

FCDA0112PCI－16MW 取扱説明書

任意波形出力機能

16MWバッファメモリ内蔵

12ビット高速D／A変換基板

対応パソコン

IBM PC/AT互換機（PCI）

2015年3月7日 ドライバーアンインストール操作説明追加
2014年10月30日発行 初版

フレックスコア

目次

本製品の使用・適用についての注意事項.....	6
故障・修理・サポート方法について	7
本製品の構成.....	7
第1章 導入・試運転.....	8
1-1 本製品の概要.....	8
1-2 ボード上の設定.....	10
1-3 入出力コネクタ・ピン接続	11
1-4 ボードのインストール.....	12
1-5 動作確認・試運転.....	12
第2章 信号入出力.....	13
2-1 アナログ入出力回路	13
2-2 アナログ出力モード・範囲	14
2-3 デジタル入出力回路	16
第3章 制御・操作.....	17
3-1 DAサンプリング動作、トリガ動作の様子.....	17
3-2 オンボードメモリの構造、動作.....	19
3-3 制御・操作の手順.....	19
即時更新モード	19
クロック同期更新・サイクルモードの場合	19
クロック同期更新・非サイクルモードの場合.....	20
3-4 クロック同期更新モードでデータを出力させる.....	20
サイクルモードと非サイクルモードについて.....	21
3-5 制御レジスタ I/O アドレスマップ	23
3-6 ボード制御部リセット及びボード番号スイッチ読み込み.....	24
3-7 DAデータコード指定.....	25
3-8 クロック源選択.....	25
3-9 読み書き対象カウンターの選択.....	26
3-10 クロック源分周比の設定.....	26
3-11 DA出力データ点数の設定	27
3-12 スケール変数 (DA) データの書き込み.....	28
3-13 アナログ出力DAデータの書き込み.....	28
3-14 トリガ及び出力更新モード指定	29
3-15 ステータス取得	30
3-16 ステータスクリア操作	30
3-17 FIFOメモリフラグ読み込み.....	31
3-18 汎用デジタル入出力	31
3-19 割り込み制御	32
3-20 SYNC・クロック出力切り替え.....	33
3-21 アナログ回路設定情報取得.....	33
3-22 アナログ回路情報設定	34
3-23 搭載メモリ量の取得.....	35
3-24 メモリ使用量の取得.....	35
3-25 複数ボードの同期運転	35
ボード上の設定	35
ボード間の接続.....	35
ソフトウェア	35

3-26 外部機器との連携動作について	36
応用例：任意波形出力及びADサンプリング	36
第4章 ソフトウェア	37
4-0 ハードウェアとソフトウェアの連携	37
4-1 本製品用ソフトウェアの種類	38
4-2 ドライバーのインストール	38
4-2-1 WindowsXPドライバーインストール及びアンインストール	38
4-2-2 Windows 7 (X86 : 32 ビット) へのインストール	42
4-2-3 Windows 7 (X64 : 64 ビット) へのインストール	45
4-3 Windows 7 ドライバーのアンインストール	48
4-4 サンプルプログラムについて	50
~~4-5 サンプルプログラム、動作確認プログラムについて (補足)~~	~~50~~
4-6 関数仕様・エラーコード	51
【1】 ボード、本ハンドラーの初期化	52
【2】 全ボードのリセット	52
【3】 トリガ関連の設定	52
【4】 波形データ転送モードの設定	52
【5】 オプション外部クロック値の設定	53
【6】 サンプリング (波形出力同期) クロック値の設定	53
【7】 スケール変数の設定	53
【8】 DLL管理バッファへのデータ転送	54
【9】 ボード内メモリへの直接データ転送	54
【10】 波形出力スタート	55
【11】 DA即時更新出力	55
【12】 ボードステータスの取得	55
【13】 波形出力の強制停止	55
【14】 本ハンドラーの終了	56
【15】 汎用デジタル出力更新	56
【16】 汎用デジタル入力	56
【17】 DAデータコード、出力モード等の設定	56
【18】 クロック / SYNC出力切換え	56
【19】 波形出力条件の再設定 (前回と同一)	57
【20】 本ハンドラーのバージョン情報取得	57
【21】 DAデータコード、出力モード等の取得	57
【22】 DAデータコード、出力モード等の設定	57
【23】 空きメモリエリア数取得	58
4-7 エラーコード	59
第5章 汎用デバイスドライバー仕様書	60
【1】 ドライバーオープン	61
【2】 ドライバースタート	61
【3】 バイトリード	61
【4】 バイトブロックリード	61
【5】 ワードリード	62
【6】 ワードブロックリード	62
【7】 ダブルワードリード	62
【8】 ダブルワードブロックリード	62
【9】 バイトライト	63

【10】 バイトブロックライト	63
【11】 ワードライト	63
【12】 ワードブロックライト	63
【13】 ダブルワードライト	64
【14】 ダブルワードブロックライト	64
【15】 ドライバーストップ	64
【16】 ドライバークローズ	64
エラーコード	65
第6章 保守、その他	66
6-1 故障・トラブル等の原因と対処	66
再点検、確認ポイント	66
ボードID	66
デジタル入出力	66
アナログ入力	66
6-2 修理のときは	66
6-3 アナログ出力範囲の再調整	67
6-4 外部制御信号、タイミング等	67

本製品の使用・適用についての注意事項

- (1) 本製品は、IBM PC/AT 互換機の PCI バス拡張 I/O スロット、または PCI バス拡張 I/O ボックスに装着して使用するものです。
- (2) 本製品が組み込まれたシステムの運用対象・方法・場所・環境等によって、故障・誤動作等が生じた場合に起こり得る、身体・生命・財産等に対する損害の回避処置は同システムの設計・製作に別途付加・反映させて下さい。本製品自体には、前述の機能はなく、従って弊社では本製品が組み込まれたシステムの運用により発生した故障・誤動作・事故に起因する身体・生命・財産等の損害に対する責任は負えません。これは本製品の故障・誤動作が原因となった場合も含み、理由の如何を問いません。
- (3) 本製品付属のソフトウェアは本製品利用の方法を示す例、またオプションの関連ソフトウェアは本製品利用の一般の便宜をはかるものであり、現在未発見のバグ存在の可能性も含めて、運用結果についての責任は一切負えません。
これらのソフトウェアには自身が組み込まれたシステムに故障・誤動作・事故等が生じた場合に起こり得る身体・生命・財産等に対する損害の回避機能はありません。御利用の場合は同システムの設計・製作で配慮・付加・反映させて下さい。
- (4) 本製品（付属ソフトウェアを含む）、及びオプションの関連ソフトウェアは医用・航空機器用・その他高信頼性・高安全性を必要とするシステムに使用しないで下さい。
- (5) 本製品付属のソフトウェアについて弊社は著作権を保持しますが、第3者の権利を侵害しない限りにおいて購入者は自身が製作するシステム等に自由に組み込み、販売する事もできます。但し、弊社製ソフトウェアのソースコードを含むソフトウェアを第3者に販売・移転するときは弊社の文書による事前許可を必要とします。
- (6) 弊社では、本製品の販売・サポート・保証の範囲を日本国内に限っています。
- (7) 本製品は改良のため仕様変更、価格改定を行うことがあります。

故障・修理・サポート方法について

- (1) 納入後 1 年間は自然故障、及び弊社製造上の問題に起因した事が明らかな故障製品に対して無償修理を行います。
- (2) 保証期間中であっても、次の場合は有償修理となります。
- ア 取扱上の不注意、誤用による故障および損傷
 - イ 弊社以外での修理、改造、分解掃除等による故障および損傷
 - ウ 泥、砂、水などのかぶり、落下、衝撃等が原因で発生した故障および損傷
 - エ 火災、地震、水害、落雷その他の天変地異、公害や異常電圧による故障および損傷
 - オ 保管上の不備（高温多湿の場所等）や手入れの不備による故障
 - カ 接続している他の機器に起因して生じた故障
 - キ その他使用者側の責に帰する故障
- (3) 修理は宅配便によるセンドバックで行います。尚運賃は互いに発送する側が負担するものとします。
出張修理は行っておりません。簡単な故障であれば一週間程度で修理・返却が可能です。故障状況によっては更に日数を要します。
- (4) 本製品使用上の質問・トラブル対応・故障修理などは入手経路の如何にかかわらず、弊社宛に直接御相談下さい。その際は、客観情報の整理・評価を行うために必ずレポートを御送付下さい。特にEmailで情報を頂く事で解決速度が大幅に向上する事が期待できます。(support@flexcore.jp) また、内容が複雑になっている場合はfax使用も有効です。
- (5) 有償修理の場合の修理費用は、基本料金¥10,000円+部品代となります。簡単な故障では凡そ2万円以下程度だと思われます。修理費用限度額がある場合は、お申し付け頂ければ超過する場合に御連絡致します。
- (6) 修理品送付先
〒301-0853
茨城県龍ヶ崎市松ヶ丘3丁目18番地3
合同会社 フレックスコア
品質管理部 林
E-mail: support@flexcore.jp
Fax:050-3488-3354

本製品の構成

本製品は FCDA0112PCI-16MWボード
入出力プラグ（アナログ、デジタル各1個）
及びドライバーソフトウェア、サンプルプログラム、取扱説明書 CDROM からなります。
取扱説明書（本文書）については、弊社ホームページよりダウンロードしていただく事も可能です。

第1章 導入・試運転

1-1 本製品の概要

本製品は任意波形出力や自動ループ制御に適したバッファメモリ付きの12ビット高速DAボードです。クロック同期出力・サイクルモードでは、処理点数の上限が65535語というハードウェア由来の制限はあるものの、予め書き込んでおいたFIFOメモリ内のDAデータ群を、最高2MHzの速度で出力させる事ができます。出力の方式としては、65535データ以内のブロックを最小1ブロック～最大65535ブロックまでの回数若しくは無限回数のループ処理を行いながら循環出力する事ができます。

出力波形1サイクルのデータ点数が上記点数以上か無限長データの場合は、FIFOメモリ空き容量のチェックを行いながらブロック転送等の処理により逐次データを補充しながら動作を行う非サイクルモードが使用できます。この場合、16M語までのデータ数であれば、無条件にFIFOメモリへデータを書き込んで動作を開始させることができます。

尚、出力波形振幅は別に用意されているスケール変数データでソフト的に、または外部アナログ入力によりハード的に制御する事も可能です。更にクロック源の入出力機能によりADボード等と組み合わせた協調制御を行うことも可能となっています。

波形DA出力機能

- ・波形DA出力 : 12ビット／1チャンネル (マスタスレーブ動作により2チャンネル以上同期動作可能)
- ・バッファメモリ : FIFO型16M語メモリ
- ・通常出力レンジ : $\pm 10\text{V} / 0 \sim +10\text{V}$ (電子スイッチ制御による切り替え)
- ・スケール変数 : 12ビット (0～4095) ソフト設定
- ・出力範囲モード : バイポーラ出力モード
: ユニポーラ出力モード
- ・出力更新モード : クロック同期 (サイクル／非サイクル) 更新、または即時更新
- ・セトリング : 1 μS (0.1%FS到達、負荷容量100pFの条件)
- ・非直線性 : 0.05%FS ($\pm 2\text{LSB}$)_{max}
- ・温度ドリフト : 20ppm/°C
- ・最大駆動負荷 : 容量1000pF以下、抵抗値5K Ω 以上

外部アナログ入力 1チャンネル装備し波形DA出力の振幅制御に使用可能 ($\pm 10\text{V}$ 以内・差動)

- ・インピーダンス : 終端10M Ω
- ・ゲイン誤差 : 最大0.05% (ゲイン=1の条件)
- ・オフセット : 最大20mV
- ・温度ドリフト : 4 $\mu\text{V} / ^\circ\text{C}_{\text{typ}}$
- ・周波数帯域 : 1.2MHz (−3dB)_{typ}

制御機能

- ・データ転送 : バイトバス制御
- ・クロック源 : 内部20MHz／16.384MHz、または外部TTL入力
- ・クロック値 : クロック源を32ビットカウンタで分周、最高動作クロックは2MHz
- ・トリガ機能 : ソフト (即スタート)、または外部TTLエッジ／レベル (帯域動作)
- ・割り込み要求要因 : クロック (波形DA出力更新タイミング)
トリガ発生
サイクルモードの指定回数終了
汎用外部割り込みTTL入力
- ・外部制御出力 : クロック (波形DA出力更新タイミング) または
SYNC (サイクルモードの先頭データ出力タイミング) 出力

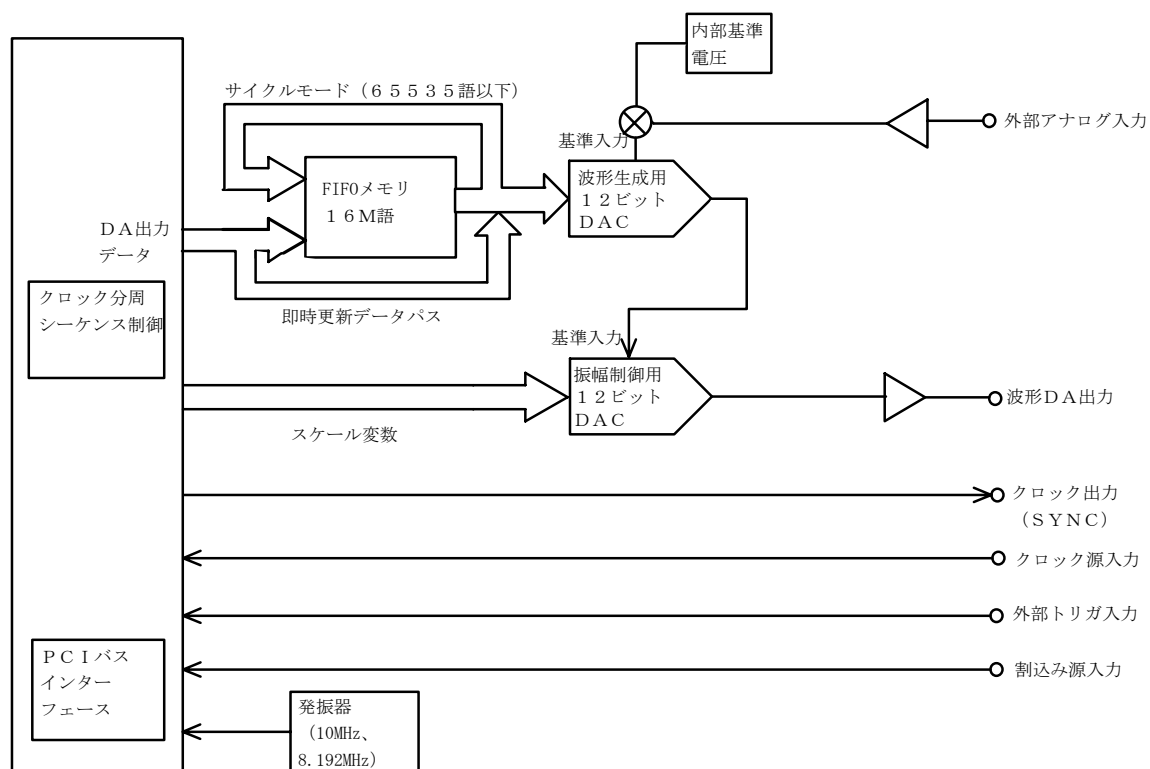


図1-1A FCDA0112PCI機能ブロック図

1-2 ボード上の設定

図1-2に本ボードの外観図を示します。

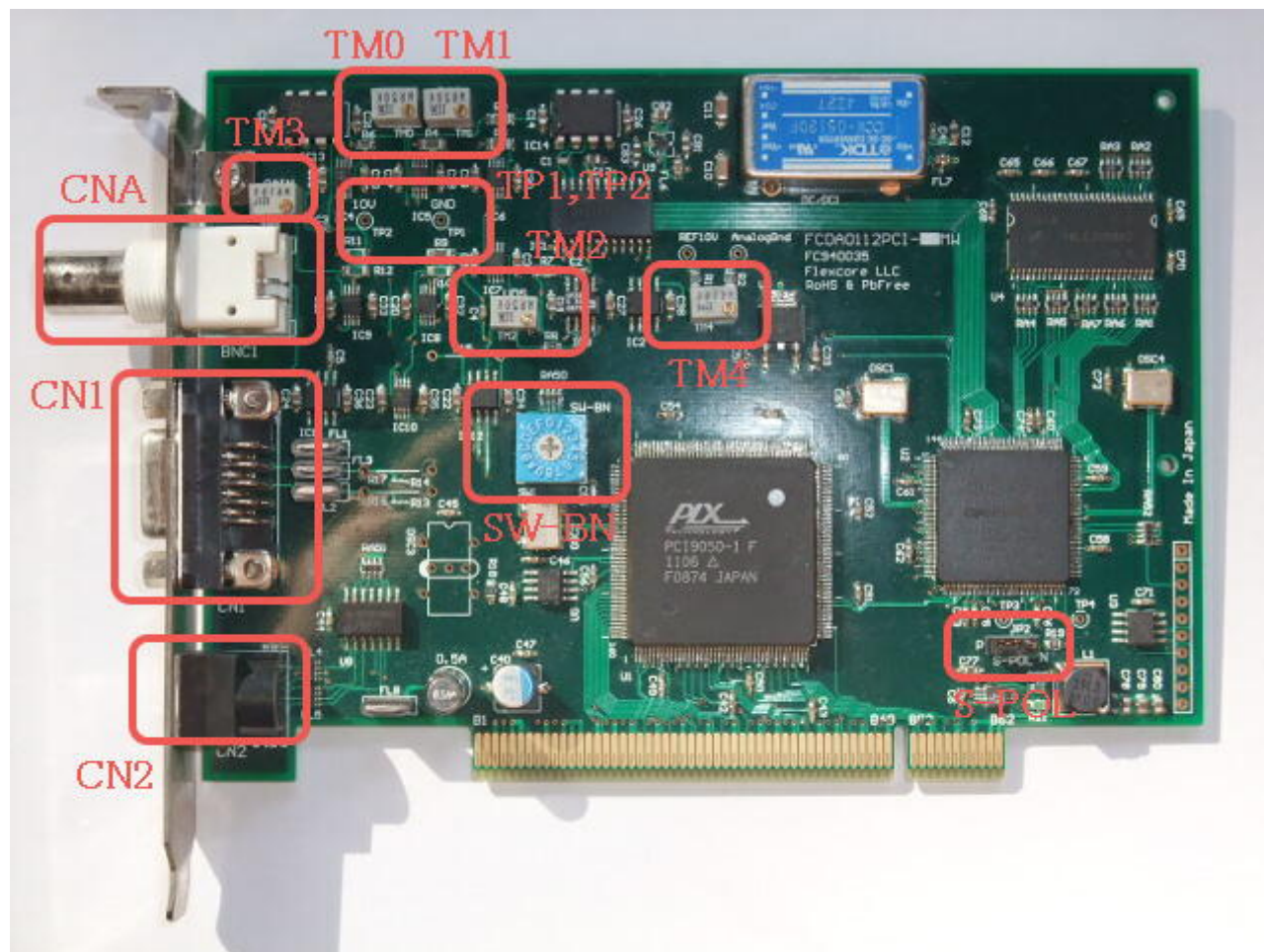


図1-2 ボード外観図

SW-BN：ボード番号設定スイッチ（出荷時：0、複数枚数使用の際はボード毎に異なる数値を設定）

S-POL：汎用デジタル出力の極性選択（出荷時：P【正論理】）

TM0：DA 出力（バイポーラ）オフセット調整用トリマ

TM1：DA 出力（ユニポーラ）オフセット調整用トリマ

TM2：ゲイン調整トリマ

TM3：振幅制御オフセット調整トリマ

TM4：基準電圧調整用トリマ

TP1,TP2：基準電圧確認用端子

CNA：アナログ出力コネクタ（BNC）

CN1：アナログ入出力コネクタ（9ピンDSUB）

CN2：デジタル入出力コネクタ（8ピン丸コネクタ：ヒロセ）

1-3 入出力コネクタ・ピン接続

本機のアナログ入出力にはD-SUBコネクタ、制御信号入出力には8ピンの丸型コネクタを使用しています。尚、アナログ出力（DA波形出力）はBNCコネクタにも接続されており、アプリケーションによって都合のよい側を選択可能です。

表1-3 入出力コネクタ

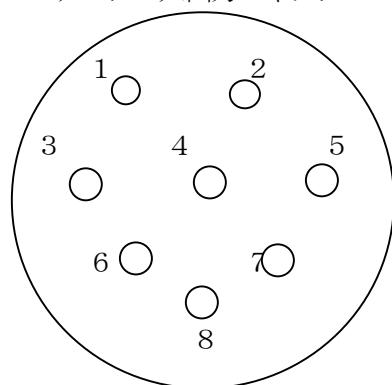
摘要	本ボード上のコネクタ	
DA アナログ出力	標準 BNC	標準 BNC
アナログ入出力	17LE-13090-27(D4AB) DDK	17JE-23090-02(D8A) 添付
制御信号入出力	HR212-10RA-8SDL(74) ヒロセ	HR212-10P-8P(71) 添付

図1-3A アナログ入出力コネクタ・ピン接続

信号名	ピン番号	ピン番号	信号名
DA アナログ出力 Vout 0	1	6	アナロググランド
空きピン	2	7	空きピン
差動アナログ入力 Vinh	3	8	アナロググランド
差動アナログ入力 Vinl	4	9	アナロググランド
空きピン	5		

注 アナロググランドとデジタルグランドはボード内部で接続されています。

図1-3B デジタル入出力コネクタ・ピン接続



添付プラグの半田付け側面視

- 1 ピン：割り込み入力
- 2 ピン：トリガ入力
- 3 ピン：クロック入力
- 4 ピン：クロック出力
- 5 ピン：汎用デジタル入力
- 6 ピン：汎用デジタル出力
- 7 ピン：+5 V出力
- 8 ピン：デジタルグランド

1-4 ボードのインストール

本製品はプラグアンドプレイに対応した PCI ボードです。そのため御使用に先立ち組み込むパソコンにインストール（認識、リソース割当て）する必要があります。そのためのステップは次の通りです。

- 1 まず、添付 CDROM よりドライバーパッケージをパソコン内部に展開します。そして、このパッケージを使用してインストールを行います。この際、本ボードはまだパソコンに取り付けてはいけません。
- 2 インストール終了後、一旦パソコンの電源を落とし、その後本ボードを PCI スロットに取り付けてからパソコンの電源を再投入します。一旦システムトレイにドライバーインストール中のメッセージが表示され、一分ほど待っていると、ドライバーがインストールされ、ハードウェアの使用準備ができた旨、表示されます。
- 3 この段階で本ボードを使用する事ができる状態になっていますので、必要に応じ適宜 CDROM 内のサンプルプログラムをパソコン内に展開して頂き実行モジュールを作成するか、或いはご自身で使用する言語システムを利用した開発を開始する事ができます。より詳細な内容については 4-1 章を参照して下さい。

1-5 動作確認・試運転

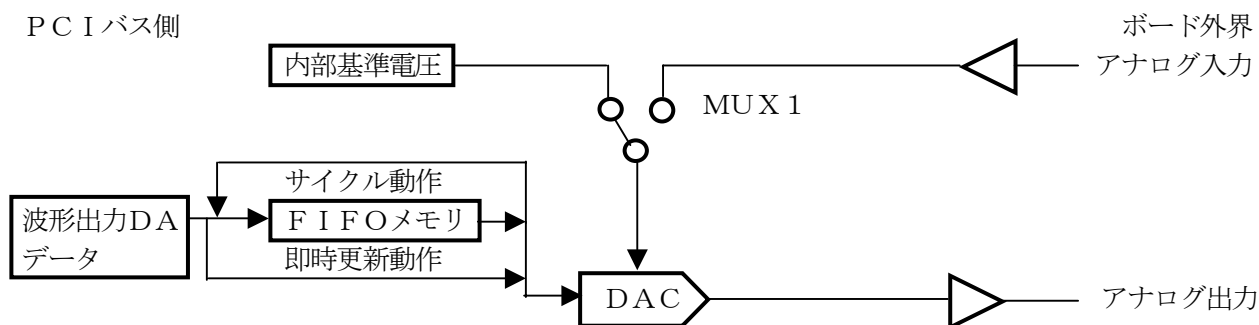
Windows7 対応の動作確認プログラムを用意しています。詳細は、弊社ホームページよりダウンロード可能なソフトウェアキットを参照してください。

第2章 信号入出力

2-1 アナログ入出力回路

本機では、1チャンネルのDA出力を最大2MHz周期で運転しており、アナログ出力の振幅制御は外部基準電圧によるハードウェア制御またはスケール変数を調整する事による内部変調によって出力を制御する事ができます。またアナログ回路の電源はロジック電源からDCDCコンバータにより±1.5Vを生成しています。

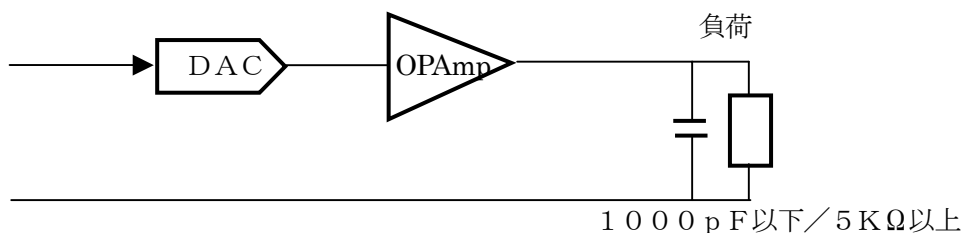
図2-1 A アナログ入出力構造



アナログ出力

オペアンプによる電圧出力です。接続できる負荷は抵抗5KΩ以上、容量負荷は最大1000pFとなっています。しかしながら、長距離をシールドケーブル等で接続する時は、シールドケーブル自身の線間容量（およそ50pF～70pF/m程の容量がある）による駆動能力オーバーに注意が必要です。

図2-1 B アナログ出力回路



アナログ入力

アナログ出力はボード上の電子スイッチMUX 1によって外部アナログ入力による振幅制御もできます。

アナログ入力回路は入力範囲±1.0V差動型です。入力端には10MΩの終端抵抗が実装されています。また、抵抗を追加する事で利得を取る事もできます。

追加抵抗とゲインとの関係は

$Gain = (49.4K\Omega / R_b) + 1$ （ R_b はゲイン設定用追加抵抗）で表されます。（オペアンプはAD620ARZを使用しています）

図2-1 C及び図2-1 Dに実際の差動信号源及びシングルエンド信号源との接続方法を示します。ここで注意して頂きたい点は、本ボードのアナログ入力が外界に対して非絶縁であるということです。そのため、グラウンドレベルの異なる信号源と接続する場合には、双方のグラウンドレベルを（回路接続により）一致させなければなりません。

特にシングルエンド信号源との接続については、システム全体のグラウンド取回しがどのようになっているのかを確認しておく必要があります。

図 2-1 C 差動信号源とのアナログ回路接続法

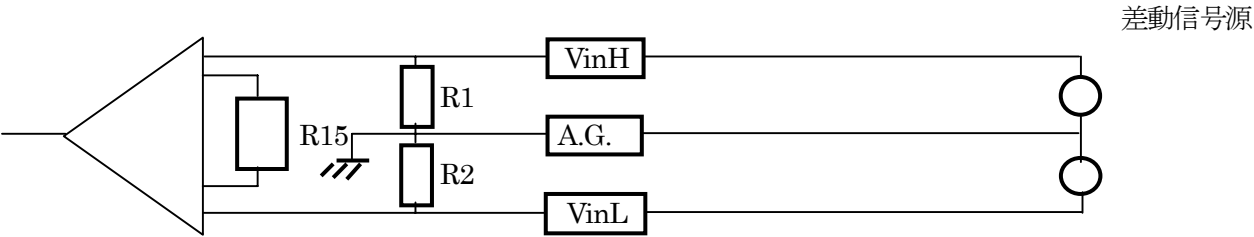
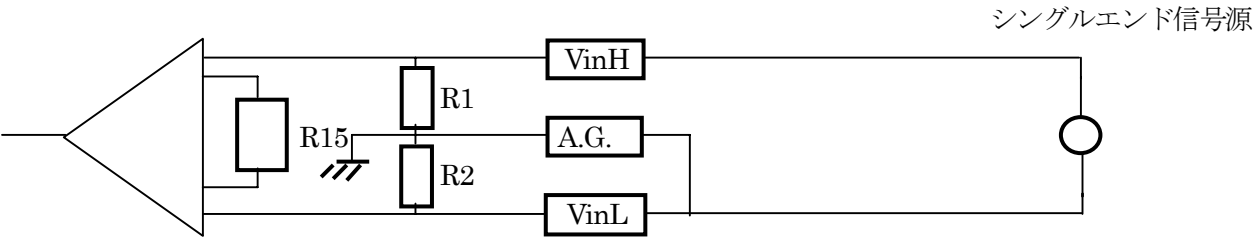


図 2-1 D シングルエンド信号源とのアナログ回路接続法



注 1 アナログ入力範囲は±10V、絶対最大定格は±15Vです。これ以上の電圧を印加すると部品破壊を含む回路故障の原因となります。

注 2 上図中R1、R2（各10MΩ）は入力オペアンプの飽和や不安定化を防止する機能があります。

注 3 通常R15は実装されていないため本回路のゲインは1倍ですが、R15を追加実装する事により、ゲインを取る事もできます。R15の値は

$R15 = 49.4\text{K}\Omega / (\text{ゲイン} - 1)$ によって求める事ができます。但し、R15の精度、温度特性等が直接ゲインに関係してくるため注意が必要です。

またR15のスペックとしてはMF1/4DCタイプ金属皮膜抵抗（KOA社製）を想定しています。

2-2 アナログ出力モード・範囲

本機のアナログ出力は、基準電圧（内部／外部）、アナログ出力DAデータの乗算値となります。出力モードはボード上のMUX（MUX2によりバイポーラ／ユニポーラ切換え、MUX3によりAモード／Bモード切換え）によって表2-2Aに示す組み合わせとなります。

表 2-2 A 各出力モードでの最大出力範囲及び分解能（基準電圧＝+10V時）

MUX 設定（ソフトウェアにより設定）			アナログ最大出力範囲 (分解能 mV/digit)	伝達関数
MUX1	MUX2	MUX3		
内部基準電圧(L)	バイポーラ(L)	Aレンジ(L)	-10.240V～+10.235V 5mV	(1)
		Bレンジ(H)	-10.000V～+9.9951V 4.88mV	
	ユニポーラ(H)	Aレンジ(L)	0～+10.2375V 2.5mV	(2)
		Bレンジ(H)	0～+9.9976V 2.44mV	
外部基準電圧(H)	バイポーラ(L)	Aレンジ(L)	-10.240V～+10.235V 5mV	(3)
		Bレンジ(H)	-10.000V～+9.9951V 4.88mV	
	ユニポーラ(H)	Aレンジ(L)	0～+10.2375V 2.5mV	(4)
		Bレンジ(H)	0～+9.9976V 2.44mV	

伝達関数

本ボードのDA出力値は

最大出力＝基準電圧×アナログ出力DAデータ

で表されます。従って、表2-2Aで示される夫々の組み合わせに従った伝達関数は以下のようになります。

伝達関数(1)：内部基準電圧を使用しバイポーラ出力モードの場合

$$V_{out} = (+10V) \times \{(n - 2048) / (Dfs \times 0.5)\}$$

伝達関数(2)：内部基準電圧を使用しユニポーラ出力モードの場合

$$V_{out} = (+10V) \times (n / Dfs)$$

ここで、

V_{out} はアナログ出力電圧

+10V は内部基準電圧

n：出力DAデータ

Dfs：フルスケール digit 値 (Aモード時：4000、Bモード時：4096)

伝達関数(3)：外部基準電圧を使用しバイポーラ出力モードの場合

$$V_{out} = (V_{in}) \times \{(n - 2048) / (Dfs \times 0.5)\}$$

伝達関数(4)：外部基準電圧を使用しユニポーラ出力モードの場合

$$V_{out} = (V_{in}) \times (n / Dfs)$$

ここで、

V_{out} はアナログ出力電圧

V_{in} は外部基準電圧

n：出力DAデータ

Dfs：フルスケール digit 値 (Aモード時：4000、Bモード時：4096)

となります。

出力DAデータと出力電圧値の関係

表2-2B、表2-2Cに出力DAデータと出力電圧値の関係を示します。(夫々Aモード時、Bモード時を示します)

表2-2B 基準電圧＝+10V時の出力DAデータ vs アナログ出力 (Aモード時)

DAデータ	アナログ最大出力範囲	
Hex/10進	±10V	0～+10V
FFF/4095	+10.235V	+10.2375V
FD0/4048	+10.000V	
FA0/4000		+10.000V
801/2049	+0.005V	
800/2048	0.000V	
7FF/2047	-0.005V	
7D0/2000		+5.000V
030/48	-10.000V	
001/1	-10.235V	+0.0025V
000/0	-10.240V	0.000V

注 上表中±10.000Vを超える値は理論値です。アナログ回路に使用されている素子の仕様により±10Vを超える値の正確度は保証されません。

表 2-2 C 基準電圧=+10V時の出力DAデータ vs アナログ出力 (Bモード時)

DAデータ	アナログ最大出力範囲	
Hex/10進	±10V	0～+10V
FFF/4095	+9.99512V	+9.99756V
FD0/4048	+9.76563V	
FA0/4000		+9.76563V
801/2049	+0.00488V	
800/2048	0.0000V	+5.0000V
7FF/2047	-0.00488V	
7D0/2000		+4.88281V
030/48	-9.76563V	
001/1	-9.99512V	+0.00244V
000/0	-10.0000V	0.0000V

2-3 デジタル入出力回路

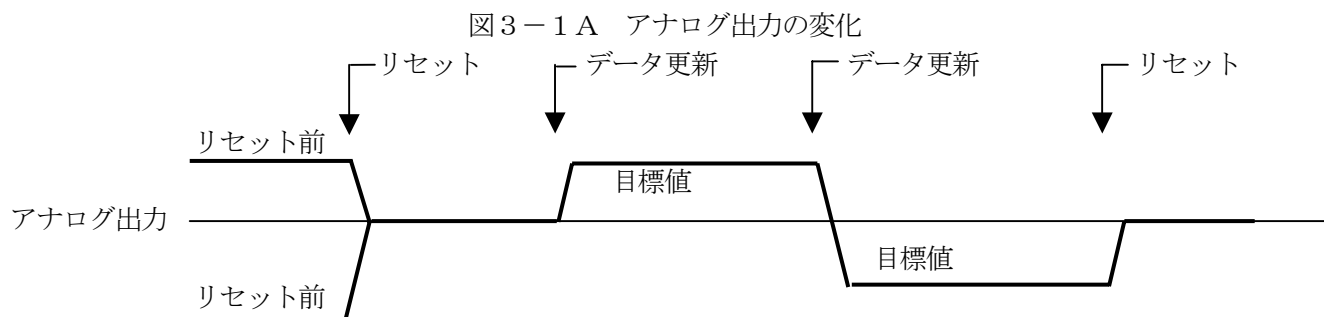
外部クロック源入力、外部トリガ入力、外部割込み入力、汎用1ビットデジタル入力は全てTTLレベルで且つ10KΩでプルアップされています。また、クロック (SYNC) 出力、汎用デジタル出力はTTLレベルです。

なお、電源投入直後の汎用1ビットデジタル出力は“0”となりますが、ソフトウェアリセット操作では変化しません。

第3章 制御・操作

3-1 DAサンプリング動作、トリガ動作の様子

電源投入、パソコン本体のリセット、或いは本ボード制御部のリセット動作直後のアナログ出力電圧は0 Vとなります。更新出力後（任意のデータがDA素子に書き込まれその出力が更新される）とアナログ出力は対応する値となり、以後次の更新までその値を保持します。



データ更新遅れ： 即時更新モード時、パソコン側からOUT命令で書き込まれるDAデータは直接DA素子のラッチに達します。この時間は数100 n S程度かかります。

また、クロック同期更新モード時、内部メモリからDA素子のラッチに転送される時間として数10 n S以下の時間が必要になります。

セトリング時間： DAデータが更新された後、実際のDA素子出力ピンのアナログ出力が指定された値に到達するまでに有限の時間が必要です。本ボードでは、10 Vの変化幅に対して0.1 %の精度で出力が確定するまでの時間（セトリング時間）として約1 μ Sが必要になります。

出力モード： ボード上のMUX（MUX 1～MUX 3）をソフトウェアで制御する事により必要なモードを指定します。

更新モード： ソフトウェア指定によりDAデータの更新タイミングを指定します。

即時更新モードは、パソコン側からの更新データが直接DA素子のラッチに書き込まれる動作です。アナログ出力データが単独に更新されます。

クロック同期更新サイクルモードは、事前にボード内のメモリ領域に書き込んでおいた波形1サイクル分のDAデータをクロックに同期して（指定回数または無限に）循環出力します。この動作はパソコン側からの制御を必要とせず、本ボード内で自動的に実行されるため最高速度でのサンプリングが可能です。一方、内部ロジックの限界として、循環出力を行う際には1サイクル間に出力できるデータ数の上限が65535データであり、連続して出力できるサイクル数の上限も65535回であるという制限事項が存在します。循環出力を行わない場合であれば、上記制限は大きく緩和され、データ点数としては16 M語-1までのデータ列を登録し出力する事ができますが、この条件は、次項で説明するクロック同期更新非サイクルモードによって実現されるものですので、当モードではサポートしていません。また、ここで使用可能データ数として-1しているのは、循環出力を行う際のハードウェア的制約であり、次項で述べる非サイクルモードでは、このような制限はありません。

クロック同期更新非サイクルモードは、ボード内のメモリ領域或いはアプリケーション側から供給されるデータをクロックに同期してDA素子に出力するモードです。（非循環）そのため、メモリ容量を上回るデータ数を出力するためには、上記何れかの方法による逐次データ補給が必要になります。

本ボードではリアルタイムにボード内メモリの空き容量を読み取る事ができますが、メモリ容量として16M語用意しており、最高サンプリング速度（2Mサンプル/秒）であっても、8秒間のデータを予めセットしておく事ができます。また、こちらのモードでは、ハードウェア上の制限事項として、データ出力用クロックの発生回数が最大で 65535×65535 という条件のみが有効になります。極論すると、全て異なるデータを 65535×65535 回出力する事もできます。しかしながら、前記内部メモリ容量を上回るデータ量をセットするためには、その補給速度が実際の最高サンプリング速度を決定してしまうため注意が必要です。

以下に、実際にクロック同期非サイクルモードで巨大データを処理した動作例を掲載しておきます。

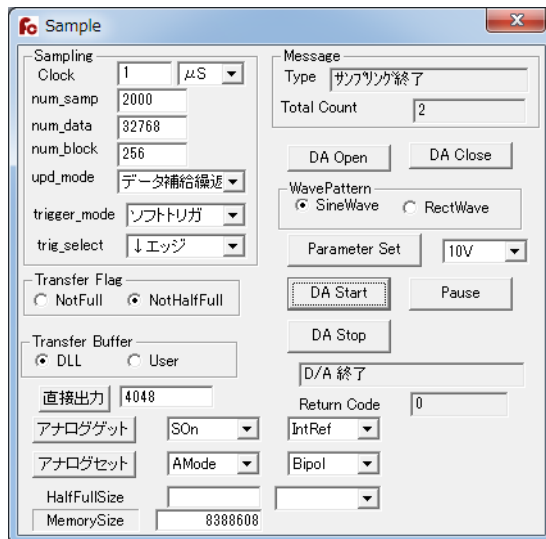


図3-1 A プログラム動作条件

プログラム動作条件は図3-1 Aに示すようになっていました。この例では、 32768×256 クロック分データが出力され、クロック周期が1 μSなので時間としては凡そ8.38秒になります。その動作結果を図3-1 Bに示します。

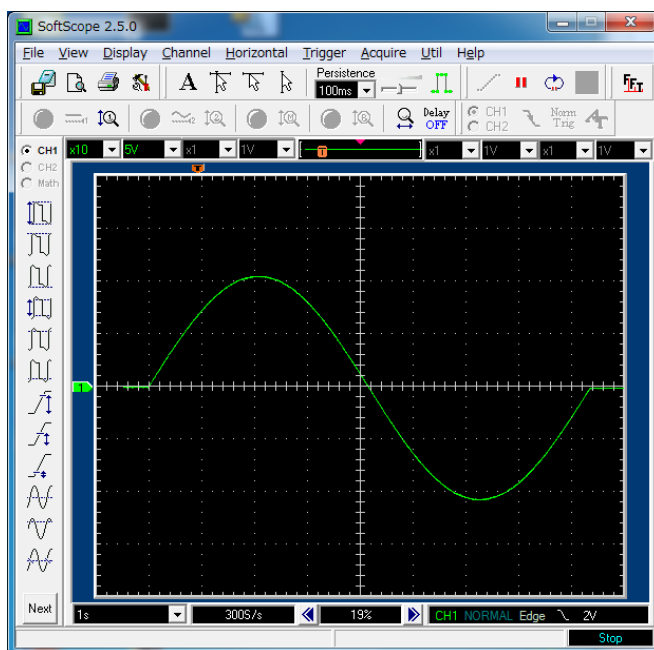


図3-1 B サンプリング結果

3-2 オンボードメモリの構造、動作

本ボードを波形出力用（クロック同期更新モード）で使用する時のDAデータは、クロックに同期して内部メモリからDA素子の内部ラッチへ自動転送される事になります。

ボード内メモリは汎用のSDRAMを使用していますが、FPGAでの制御により等価的に巨大FIFOとして使用できるようになっています。動作としてFIFOメモリと等価ですので、その読み書き動作は非同期で同時であっても問題ありません。即ち追加データを書き込む際出口側のタイミングを考慮する必要がありません。

図3-2 ボード内部メモリ内のDAデータ（FIFOワードイメージ）

メモリ内ロケーション	2n+4	2n+3	2n+2	2n+1	2n	2n-1	7	6	5	4	3	2	1	0
DAデータブロック	2n+4	2n+3	2n+2	2n+1	2n		7	6	5	4	3	2	1	0

最古データ

本ボードでは、FIFOメモリの代わりとして前述の巨大FIFOイメージメモリを使用しています。そのため、通常のFIFOで存在したハーフフルフラグは本来の意味とは異なり、使い勝手があまりよくありません。そのため、本ボードでは、FIFOメモリの使用量をアプリケーション側から任意のタイミングで確認できるようなレジスタセットに拡張機能が割り当てられています。（データあり：ノットエンプティとデータ満杯：データフルは定義されソフトウェアで読み取る事も可能です。）

3-3 制御・操作の手順

次項以下で各制御レジスタの機能を説明するに先立ち、操作手順の概要を示します。
ここでは、疑似C言語により割り込みは使用しない形でコーディングを行っています。

即時更新モード

（全ての入出力単位はバイト：8ビットです）

- | | | |
|---|-----------------------------|------------------------|
| 1 | r s t = i n p (BASE+0 x 07) | // ボードリセット |
| 2 | o u t p (BASE+0 x 2, 0) | // DAデータコードをバイナリモードに指定 |
| 3 | o u t p (BASE+0 x 3, 0) | // 即時更新モードを指定 |
| 4 | o u t p (BASE+0 x D, BL) | // スケール変数データ下位バイト |
| | o u t p (BASE+0 x D, BH) | // スケール変数データ上位バイト |
| 5 | o u t p (BASE+0 x C, AL) | // DA出力コード下位バイトセット |
| | o u t p (BASE+0 x C, AH) | // DA出力コード上位バイトセット |

クロック同期更新・サイクルモードの場合

- | | | |
|---|---------------------------------|------------------------|
| 1 | r s t = i n p (BASE+0 x 07) | // ボードリセット |
| 2 | o u t p (BASE+0 x 2, 0) | // DAデータコードをバイナリモードに指定 |
| 3 | o u t p (BASE+0 x 8, C k S e 1) | // クロック源選択 |

クロック分周比をセットする

- | | | |
|----|---------------------------------|----------------------|
| 4 | o u t p (BASE+0 x 7, 3) | // 素子1の制御レジスタを指定 |
| 5 | o u t p (BASE+0 x 6, 0 x 36) | // 制御データ（固定値） |
| 6 | o u t p (BASE+0 x 7, 0) | // 素子1のカウンター0を指定 |
| 7 | o u t p (BASE+0 x 6, D i v L 0) | // 分周比0（下位バイト）セット |
| 8 | o u t p (BASE+0 x 6, D i v H 0) | // 分周比0（上位バイト）セット |
| 9 | o u t p (BASE+0 x 7, 3) | // 素子1の制御レジスタを指定 |
| 10 | o u t p (BASE+0 x 6, 0 x 76) | // 制御データ（固定値） |
| 11 | o u t p (BASE+0 x 7, 1) | // 素子1のカウンター1を指定 |
| 12 | o u t p (BASE+0 x 6, D i v L 1) | // 分周比1（下位バイト）コードセット |
| 13 | o u t p (BASE+0 x 6, D i v H 1) | // 分周比1（上位バイト）コードセット |

ブロックデータ点数、ブロック点数をセットする

14	o u t p (BASE+0 x 7, 7)	// 素子2の制御レジスタを指定
15	o u t p (BASE+0 x 6, 0 x 3 4)	// 制御データ (固定値)
16	o u t p (BASE+0 x 7, 4)	// 素子2のカウンター0を指定
17	o u t p (BASE+0 x 6, NUML)	// ブロックデータ点数下位バイトを出力
18	o u t p (BASE+0 x 6, NUMH)	// ブロックデータ点数上位バイトを出力
19	o u t p (BASE+0 x 7, 7)	// 素子2の制御レジスタを指定
20	o u t p (BASE+0 x 6, 7 2)	// 制御データ (固定値)
21	o u t p (BASE+0 x 7, 5)	// 素子2のカウンター1を指定
22	o u t p (BASE+0 x 6, BLKL)	// ブロック数下位バイトを出力
23	o u t p (BASE+0 x 6, BLKH)	// ブロック数上位バイトを出力

スケールDAC設定

24	o u t p (BASE+0 x D, SCALEL)	// スケール変数下位バイト書き込み
25	o u t p (BASE+0 x D, SCALEH)	// スケール変数上位バイト書き込み
26	o u t p (BASE+0 x 3, 1)	// クロック同期更新モードを指定
27	以下、出力するDAデータを下位、上位の順に出力	
	.	.
xx	o u t p (BASE+0 x 0, Chnl)	// n 番目 DA 出力下位バイトセット
xx+1	o u t p (BASE+0 x 0, Chnh)	// n 番目 DA 出力上位バイトセット
	o u t p (BASE+0 x 3, 0 x 8 5)	// トリガ指定 (ソフトトリガ)
	o u t p (BASE+0 x 3, 5)	// 停止 (=ソフトトリガ禁止)

注1 17、18 で指定したブロック内データ点数と、メモリに書き込む点数nが一致する事

注2 DAデータ書き込み前に 26 のクロック同期更新モード指定を行っておく事

注3 トリガ指定モードとして0 x 8 Dを出力した場合は、指定ブロック数出力により動作は停止するのでプログラムによる停止動作 (0 x cを出力) を行う必要はありません。その代り、トリガ許可ビット (ビット7及びビット6) が動作停止と同時に無効になるため、再動作を行わせるためには、これらビットを、一旦クリアしておく必要があります。

クロック同期更新・非サイクルモードの場合

この場合は、最後にトリガ出力として

o u t p (BASE+0 x 3, 0 x 8 1) とし

o u t p (BASE+0 x 3, 0 x 1) とする事で動作を停止させます。

また、トリガ指定モードとして0 x 8 9を出力した場合は、指定ブロック数出力により動作は停止するので、プログラムによる停止動作 (0 x 9を出力) を行う必要はありませんが、前頁で示したトリガ許可ビットの無効化が発生するため同じように一旦これらのビットをクリアしておかないと再起動ができません。

また、非サイクルモードでは、F I F Oメモリの内容を消費しながらの動作となるため、連続動作を行わせるためには、内部メモリのステータス等を監視しながら、逐次出力データを追加出力しなくてはなりません。

この際、追加書き込み操作が間に合わないと同期更新エラーとなり動作が停止してしまうため注意が必要です。

3-4 クロック同期更新モードでデータを出させる

本ボードのシーケンス制御は、基本的に2本の16ビットカウンターの組み合わせにより、最大で65535個のクロックを、最大65535回繰り返して発生させるというものです。この動作はサイクルモードであっても、非サイクルモードであっても変わりません。そのため、例えば32768回のクロックを256回繰り返す設定にすると合計で8M回のクロック出力となります。この時、本ボード内部のF I F Oメモリに8M語のデータを予め書き込んでおくと、このデータストリームが連続して出力される、ということになります。しかしながら、現在のハンドラーとの関係で、この機能を実現する事はかなり困難です。サンプリングスタート機能は以下

のように定義されています。

```
int __stdcall DA781_Start_Samp( WORD num_data, WORD num_block, int upd_mode );
```

ここで、第一引数 `num_data` と第二引数 `num_block` は夫々WORD 引数であるため、それらの最大値は 65535 を超える事はできません。そうすると、ハンドラー側としては、クロック同期更新モードで動作が指示された場合、これまで利用していたサイクルモードと、そうではない非サイクルモードとを区別する情報を得る事はできません。

そのため、6 5 5 3 5 データを上回るデータ処理は、非サイクル動作モードで対応する事とします。

その方策は以下の通りです。

まず、サンプリング回数カウンターとサイクルカウンタは例の通り夫々 3 2 7 6 8 及び 5 1 2 とします。

データ生成については、非サイクルモードという事で、データ長を 1 6 M 語 (1 6 7 7 7 2 1 6) として生成を行います。

動作モードは、クロック同期更新・非サイクルモードとします。

D 1 1 バッファを使用する場合は `DA781_Write_DLLBuf` 関数において、1 6 7 7 7 2 1 6 (非同期モードとして) `num_data` を 3 2 7 6 8 として関数を呼び出します。

ユーザーバッファを使用する場合は、`DA781_Write_DirectFifo` 関数において `num_data` を 1 6 7 7 7 2 1 6 として関数を呼び出します。(こちらの場合は非同期モードのみなのでデータ数はこれ一種類で済みます。)

その後、サンプリングスタートを実行させると、FIFO メモリから出力データを取り出しながらDA出力を更新していく事ができます。

サイクルモードと非サイクルモードについて

本ボードでは、クロック同期出力モードとして、サイクルモード(循環モード)と非サイクルモード(データ追加型出力モード)の2種類をサポートしています。これらのモードは互いに排他な関係にあるため、使用に際しては、どちらか一方での動作を行うことになります。サイクルモードでは、FIFO出力を再帰的にFIFO入力として使用するため、その動作中に新しいデータを追記する事はできません。従って、どのような形であれ、上記数値(6 5 5 3 5 データ)を上回るデータを処理する事は不可能です。また、その動作形態上、サンプリング動作を開始する前に全てのデータをFIFO内部にセットしておかなければなりません。

他方、非サイクルモードでは、これを上回るデータ長を扱う事ができますが、サンプリング動作中に逐次データを追加する必要があるため、データ追加速度がサンプリング速度を下回ってしまった場合、最終的にデータアンダーランエラーを発生しサンプリング動作が中断してしまう事になります。その際のバッファリングについては、D 1 1 内部のバッファを使用するか、或いはユーザープログラムから直接出力するかの何れかの方法を使用することになりますが、総合的にどちらのパターンが性能的に優位にあるかを実際のプログラムによって評価し、決定する事が重要だと考えられます。

表 3-3 に、これらの動作モード間の比較を纏めています。(次ページ)

動作モード	動作内容	使用バッファ	使用上の注意
サイクル	num_data 数のデータを連続出力 (num_data<65536 である必要がある)	ユーザーバッファ	サンプリング動作前に全データを FIFO メモリへ書き込む必要がある (65535個以上のデータを書き込んでも、オーバーする部分は無視される)
		D11 バッファ	サンプリング動作前に全データを FIFO メモリへ書き込む必要がある (65535個以上のデータを書き込んでも、オーバーする部分は無視される)
サイクルブロック	num_data 数× num_block ブロックを連続出力 (循環出力では num_data<65536 である必要がある)	ユーザーバッファ	サンプリング動作前に全データを FIFO メモリへ書き込む必要がある (65535個以上のデータを書き込んでも、オーバーする部分は無視される)
		D11 バッファ	サンプリング動作前に全データを FIFO メモリへ書き込む必要がある (65535個以上のデータを書き込んでも、オーバーする部分は無視される)
非サイクルブロック	num_data 数× num_block ブロックを連続出力	ユーザーバッファ	サンプリング前に全データを書き込まなくても動作できる。FIFO メモリへの追加データはアプリ側で処理する必要がある
		D11 バッファ	サンプリング前に全データを D11 バッファへ書き込む必要がある。FIFO メモリへの追加データはハンドラー内部で処理される
非サイクル	追記データを連続出力	ユーザーバッファ	サンプリング前に全データを書き込まなくても動作できる。FIFO メモリへの追加データはアプリ側で処理する必要がある
		D11 バッファ	サンプリング前に全データを D11 バッファへ書き込む必要がある。FIFO メモリへの追加データはハンドラー内部で処理される

表 3-3 動作モードの比較

3-5 制御レジスタ I/O アドレスマップ

表 3-5 に本ボード上の制御レジスタの I/O アドレスを記します。表中の【BASE】はプラグアンドプレイで設定される I/O ベースアドレス値です。

表 3-5 制御レジスタの I/O アドレス

I/O アドレス	IN/OUT	制御レジスタ名・機能	記載ページ
【BASE】+F	IN		
	OUT		
【BASE】+E	IN		
	OUT		
【BASE】+D	IN		
	OUT	スケール変数 DA データ書き込み	3-12 項
【BASE】+C	IN	アナログ回路設定情報読み込み (注 1)	3-21 項
	OUT	アナログ出力 DA データ書き込み	3-13 項
【BASE】+B	IN	ボード番号スイッチ読み込み	3-6 項
	OUT	アナログ回路設定情報書き込み (注 1)	3-22 項
【BASE】+A	IN	汎用デジタル入力	3-18 項
	OUT	汎用デジタル出力	3-18 項
【BASE】+9	IN	搭載メモリサイズ取得 (注 1)	3-23 項
	OUT	外部出力選択 (クロックまたは SYNC)	3-20 項
【BASE】+8	IN	FIFO メモリ保持データ数読み込み (注 1)	3-24 項
	OUT	クロック源選択	3-8 項
【BASE】+7	IN	ボードリセット	3-6 項
	OUT	読み書き対象カウンタ (内部レジスタ) 選択	3-9 項
【BASE】+6	IN	カウンタ (内部レジスタ) 読出し	3-10,11 項
	OUT	カウンタ (内部レジスタ) 書き込み	3-10,11 項
【BASE】+5	IN	ボードステータス読み込み	3-15 項
	OUT	ボードステータスクリア	3-16 項
【BASE】+4	IN	FIFO メモリフラグ読み込み	3-17 項
	OUT	割り込み制御 (要因指定)	3-19 項
【BASE】+3	IN		
	OUT	トリガ・出力更新モード指定	3-14 項
【BASE】+2	IN		
	OUT	DA データコード指定	3-7 項
【BASE】+1	IN		
	OUT	割り込み要求セット・クリア	3-19 項
【BASE】+0	IN		
	OUT		

全てのレジスタは 8 ビット長でのアクセスとなります。

注 1 本機能は FCDA0112PCI で導入されたため、マイクロサイエンス製ドライバー (H781_2K.DLL) では操作出来ません。(汎用 I/O ハンドラーでは操作可能です)

3-6 ボード制御部リセット及びボード番号スイッチ読み込み

$rst = inp (BASE + 7);$

本ボード制御部全体を初期化します。本操作で読み込んだデータは、このボードのIDです。本操作は電源オンまたはパソコン本体のハードウェアリセットと等価の機能ですが、汎用デジタル出力は変化せず保持されます。処理内容は

- ・ 制御部内各制御レジスタの初期化
- ・ ボードステータスの初期化
- ・ サンプリング動作の停止
- ・ アナログ出力は0V
- ・ FIFOメモリ内容のクリア

を行います。

表3-6A 【BASE+7】入力ポートの構成

ビット	各ビットの機能・意味
B7 B6 B5 B4 B3 B2 B1 B0	FCDA0112PCI-16MWのボードID=12H

本ボードを複数使用する場合、または専用ドライバーで動作を行わせる場合、マスター、スレーブをボード番号スイッチの値を使用して切り分けます。汎用ドライバーを使用する限りにおいてはこの機能は必ずしも必要ではありませんが、この機能を利用してマスター、スレーブを切り分ける事により、PCIスロットの位置と無関係にボードの処理順序を規定する事ができるため、有用だと思われます。

$rst = inp (BASE + B);$

表3-6B 【BASE+B】入力ポートの構成

ビット	各ビットの機能・意味
B7 ～B4	未使用 (0)
B3 B2 B1 B0	ボード番号スイッチの値 (0～FH)

3-7 DAデータコード指定

out (BASE+2, code);

DAデータコードその他のアナログ回路設定条件を指定します。

表3-7 【BASE+2】入出力ポートの構成

ビット	機能	= 1 の時	= 0 の時	リセット時
B 7 ~ B 1	未使用			
B 0	DAデータコード指定	2の補数	オフセットバイナリ	0

3-8 クロック源選択

out (BASE+8, clk sel);

ここで選択されたクロック源信号は、次項で説明するプログラマブルカウンタにより分周され、目的のサンプリング（同期DA出力タイミング）クロックとなります。

表3-8 A 【BASE+8】出力ポートの構成

ビット	各ビットの機能・意味	= 1 の時	= 0 の時	リセット時
B 7	外部クロック源の有効極性指定	↑ (+)	↓ (-)	0
B 6 B 5	内部クロック源の選択データ	表3-8 B参照		0 0
B 4	クロック源選択（外部・内部）	外部	内部	0
B 3 ~ B 0	未使用			

表3-8 B 内部クロック選択パラメータ表

ビット		選択される内部クロック源	
B 6	B 5	クロック源周波数	主な用途
1	1	無効	
1	0	オプション	
0	1	8.192MHz	周波数解析
0	0	10.0MHz	汎用計測

3-9 読み書き対象カウンターの選択

out (BASE+0x7, ctr);

この後に読み書きするカウンター、またはカウンター制御レジスタを選択します。

カウンター素子は2個あり各々16ビット×2本構成です。実際にはこれらカウンターは論理デバイスでFPGA内部にロジックの形で実現されています。

素子1はクロック源分周用に、素子2はクロック同期出力時のDAデータ点数カウントに使用しています。

表3-9A【BASE+0x7】出力ポートの構成

ビット	各ビットの機能・意味	リセット時
B7～B3		
B2～B0	カウンターまたは制御レジスタ選択データ (表3-8B参照)	0

表3-9Bカウンターまたは制御レジスタ選択データ

B2	B1	B0	選択されるカウンターまたは制御レジスタ
1	1	1	素子2: 制御レジスタ: 1バイト
1	1	0	素子2: カウンター#2: 不使用
1	0	1	素子2: カウンター#1: ブロック数カウント (2バイト)
1	0	0	素子2: カウンター#0: データ点数カウント (2バイト)
0	1	1	素子1: 制御レジスタ: 1バイト
0	1	0	素子1: カウンター#2: 不使用
0	0	1	素子1: カウンター#1: クロック源分周上位ワード
0	0	0	素子1: カウンター#0: クロック源分周下位ワード

3-10 クロック源分周比の設定

クロック源を分周するためには、前項説明中素子1カウンターを使用します。その操作手順は次の通りです。

outp (BASE+7, 0x3); //素子1の制御レジスタを選択

outp (BASE+6, 0x36); //制御データ書き込み (固定値)

outp (BASE+7, 0x0); //素子1のカウンター#0を対象に指定

outp (BASE+6, DV0L); //下位バイト書き込み

outp (BASE+6, DV0H); //上位バイト書き込み

outp (BASE+7, 0x3); //素子1の制御レジスタを選択

outp (BASE+6, 0x76); //制御データ書き込み (固定値)

outp (BASE+7, 0x1); //素子1のカウンター#1を対象に指定

outp (BASE+6, DV1L); //下位バイト書き込み

outp (BASE+6, DV1H); //上位バイト書き込み

この結果、出力されるクロック周期Tは

$T = 125nS \times (DV0H \times 256 + DV0L) \times (DV1H \times 256 + DV1L)$ によって求める事ができます。

3-11 DA出力データ点数の設定

DA出力データ点数及びブロック数の制御は、3-9項で説明したカウンターの内素子2を使用して行います。具体的には、素子2のカウンター#0には1ブロック当りのデータ点数(2~65535)を素子2のカウンター#1には、指定ブロック数(1~65535)をセットします。注意点としては、1ブロック当りのデータ点数の最小値は1ではなく2である事です。次に実際の制御方法について説明します。

```

o u t p ( B A S E + 7 , 0 x 7 ) ; //素子2の制御レジスタを対象に指定
o u t p ( B A S E + 6 , 0 x 3 4 ) ; //制御データ (固定値)
o u t p ( B A S E + 7 , 0 x 4 ) ; //素子2のカウンター#0を対象に指定
o u t p ( B A S E + 6 , N U M L ) ; //下位バイト書き込み
o u t p ( B A S E + 6 , N U M H ) ; //上位バイト書き込み
o u t p ( B A S E + 7 , 0 x 7 ) ; //素子2の制御レジスタを対象に指定
o u t p ( B A S E + 6 , 0 x 7 2 ) ; //制御データ (固定値)
o u t p ( B A S E + 7 , 0 x 5 ) ; //素子2のカウンター#1を対象に指定
o u t p ( B A S E + 6 , B L K L ) ; //下位バイト書き込み
o u t p ( B A S E + 6 , B L K H ) ; //上位バイト書き込み

```

この例ではNUMH×256+NUML個のデータブロックをBLKH×256+BLKL回繰り返し出力するという制御を指定したことになります。

出力済みデータ点数の確認 (データ点数の読み)

クロック同期更新モードで出力中、素子2のカウンター#0、カウンター#1の現在値を読み込む事により、現在までの出力済み点数を知る事ができます。

動作開始時のカウンター#0設定値をNUM (1ブロック当りのデータ点数)

カウンター#1設定値をBLK (指定ブロック数) とし

現在のカウンター#0読み込み値CNM

カウンター#1読み込み値CBKとすると、これまでに出力されたデータ点数DOUTは

$$DOUT = (BLK - CBK) \times NUM + (NUM - CNM)$$

で求める事ができます。

ここで、CNM、CBKの読み込み方法は以下に示す順序で行います。

```

o u t p ( B A S E + 7 , 0 x 7 ) ; //素子2の制御レジスタを対象に指定
o u t p ( B A S E + 6 , 0 x D 6 ) ; //制御データ (固定値)
o u t p ( B A S E + 7 , 0 x 4 ) ; //素子2のカウンター#0を対象に指定
C N M L = i n p ( B A S E + 6 ) ; //下位バイト読み込み
C N M H = i n p ( B A S E + 6 ) ; //上位バイト読み込み
o u t p ( B A S E + 7 , 0 x 5 ) ; //素子2のカウンター#1を対象に指定
C B K L = i n p ( B A S E + 6 ) ; //下位バイト読み込み
C B K H = i n p ( B A S E + 6 ) ; //上位バイト読み込み

```

実際には、この素子に対しては、その動作を制御するために別系統からの操作も行われている関係で、アクセスに対する排他制御を確立しておかないと、異常な動作をしてしまう可能性も考えられるため十分な検討が必要です。

3-12 スケール変数(DA)データの書き込み

o u t p (BASE+0xD, SL); //スケール変数DAデータ下位バイト書き込み

o u t p (BASE+0xD, SH); //スケール変数DAデータ上位バイト書き込み

これらのデータは実DAチップへ出力されます。この結果、実際のDA出力は次のようになります。

バイポーラ出力時

$$DA_{out} = V_{in} \times \{(n - 2048) / (Dfs \times 0.5)\} \times (Nr / Dfs)$$

ユニポーラ出力時

$$DA_{out} = V_{in} \times (n / Dfs) \times (Nr / Dfs)$$

但し

V_{in} : 基準電圧 Nr : スケール変数用DAデータ n : アナログ出力用DAデータ Dfs : フルスケール digit 値 (Aモード時: 4000、Bモード時: 4096)
--

3-13 アナログ出力DAデータの書き込み

アナログ出力DAデータを書き込む前に次項で説明するアナログ出力更新モード選択を行ってください。

(クロック同期更新モード時は1を、即時更新モード時は0をセットする)

この事により、書き込まれたアナログ出力DAデータはクロック同期更新モード時オンボードメモリに、即時更新モード時には、DA出力素子に直接書き込まれます。

o u t p (BASE+0xC, RL); //下位バイト書き込み

o u t p (BASE+0xC, RH); //上位バイト書き込み

実際のアナログ出力データは、以下に示すような計算式によって求められます。

バイポーラ出力時

$$DA_{out} = V_{in} \times \{(n - 2048) / (Dfs \times 0.5)\} \times (Nr / Dfs)$$

ユニポーラ出力時

$$DA_{out} = V_{in} \times (n / Dfs) \times (Nr / Dfs)$$

但し

V_{in} : 基準電圧 Nr : スケール変数用DAデータ n : アナログ出力用DAデータ Dfs : フルスケール digit 値 (Aモード時: 4000、Bモード時: 4096)
--

3-14 トリガ及び出力更新モード指定

o u t p (BASE+3, t u m) ;

アナログ出力の更新モード選択及びトリガ制御（クロックのスタート/ストップ）を行います。波形出力動作を行う場合、アナログ出力DAデータ書き込み前にクロック同期更新モードに設定しておかないとデータがオンボードメモリに書き込まれないので注意が必要です。

表3-14A 【BASE+3】出力ポートの構成

ビット	各ビットの機能・意味	=1の時	=0の時	リセット時
B 7	ソフトトリガ	許可（即スタート）	禁止（ストップ）	0
B 6	外部トリガ制御	許可（入力待ち）	禁止（ストップ）	0
B 5	外部トリガ信号の有効極性	+ (↑)	- (↓)	0
B 4	外部トリガ動作形態	エッジ	レベル（帯域動作）	0
B 3	動作終了条件（クロック同期時）	指定ブロック数出力	無制限	0
B 2	F I F Oメモリ動作モード	サイクル（循環）	非サイクル	0
B 1	未使用			0
B 0	アナログ出力更新モード	クロック同期更新	即時更新	0

補足説明

B 7 : セットする事でプログラム上任意のタイミングでクロックを即スタートさせます。

B 6 : セットする事で外部トリガTTL信号の有効極性（エッジまたはレベル）入力待ちになります。

B 4 : レベル（帯域動作）を指定すると有効レベル時のみクロックが動作する特殊なトリガ形態となります。

B 3 : 無制限を選択すると、強制停止操作を行うまでクロックが動作し続けるモードになります。指定ブロック数による終了の場合は、終了時点でB 7, B 6各ビットは無効になります。再スタート時にはこれらを一旦クリアし再セットする必要があります。

B 2 : サイクル動作では、メモリ内のデータ数を1周期として循環出力します。非サイクル動作では更新出力されたデータ分だけメモリ内部のデータが減少してゆくため、継続動作のためには逐次データの補充が必要になります。

B 0 : アナログ出力DAデータの更新モード選択ビットです。マニュアル操作による単純なDA出力なら即時更新モード、指定クロックに同期して自動出力する時はクロック同期更新モードを選択します。

更新モード ビット2からビット0で指定される動作形態の組み合わせを表3-13Bに記します。

表3-14B 各種更新モード

B 2	B 1	B 0	アナログ出力	内部基準電圧
1	×	1	①クロック同期更新・サイクルモード	即時更新動作
0	×	1	②クロック同期更新・非サイクルモード	
×	×	0	③即時更新モード	

①、②はクロックに同期してF I F Oメモリ内のデータが更新出力されます。

③はDAデータが直接出力素子に書き込まれます。

① のクロック同期更新・サイクルモードで出力中はF I F Oメモリに追加データの書き込みができません。

3-15 ステータス取得

`rst = inp (BASE + 5);`

本ボードのDAデータ出力、クロック、割り込み要求、FIFOメモリの使用状況等動作状態を取得します。

表3-15A【BASE+5】入力ポートの構成

ビット	各ビットの機能・意味	= 1 の時	= 0 の時	リセット時
B 7	指定ブロック数出力	終了	出力中 (開始前)	0
B 6	割り込み要求	発生	未発生	0
B 5	クロック有効エッジ (↓)	発生	未発生	0
B 4	ブロック先頭位置 (SYNC)	発生	未発生	0
B 3	トリガ発生	発生	未発生	0
B 2	クロック同期更新エラー	発生	未発生	0
B 1	ノットハーフフル	ハーフ以下	ハーフ超	1
B 0	ノットフル	空きあり	空きなし (満杯)	1

補足説明

ビットB 3以外は次ページのステータスクリア操作によってクリアされます。

B 7 : 指定ブロック数出力によりセットされます

B 6 : 各種割り込み要因によりセットされます

B 5 : クロックの有効エッジ (↓) によりセットされます

B 4 : ブロック先頭信号 (SYNC) によりセットされます

B 3 : トリガの発生によりセットされトリガ制御ビットのクリアによりリセットされます

B 2 : FIFOメモリの入出力エラーを検出するとセットされます。具体的にはアンダーラン即ちFIFOメモリが空の状態でのクロック有効エッジタイミングが発生した時若しくはオーバーラン即ちFIFOメモリが満杯の状態でも更にDA出力データを書き込もうとした時にセットされます。

B 1 : ノットハーフフル即ち内部メモリに指定サイズ以下のデータしか残っていない条件でセットされます。

B 0 : ノットフル即ち内部メモリに空きエリアがある場合セットされます。

3-16 ステータスクリア操作

ステータスレジスタの内特定ビットに対しては、ソフトウェア操作によってビット単位でのクリア操作ができます。

表3-16【BASE+5】出力ポートの構成

ビット	各ビットの機能・意味	= 1 の時	= 0 の時	リセット時
B 7	指定ブロック数出力	クリアする	クリアしない	0
B 6	割り込み要求	クリアする	クリアしない	0
B 5	クロック有効エッジ (↓)	クリアする	クリアしない	0
B 4	ブロック先頭位置	クリアする	クリアしない	0
B 3	未使用			0
B 2	クロック同期更新エラー	クリアする	クリアしない	0
B 1	未使用			0
B 0	FIFOメモリ単体のリセット	リセットする	リセットしない	0

3-17 FIFO メモリフラグ読み

$fsts = inp(BASE + 4);$

FIFOメモリの状態を示すステータスフラグを取得します。このレジスタから読み出すステータスにより、FIFOの動作状態を調べ、必要に応じて追加データを書き込みます。特に非循環型の連続動作時には、アプリケーションからFIFOのステータスを確認しながら、データを払い出す必要があります。

ポートの構成は表3-17に示す通りです。

表3-17 【BASE+4】入力ポートの構成

ビット	ビットの意味・機能	=1の時	=0の時	リセット時
B7 ~ B3	未使用			
B2	ノットフル	空きあり	満杯	1
B1	ノットハーフフル	1/2以下	1/2超え	1
B0	ノットエンプティ	データあり	データなし	0

3-18 汎用デジタル入出力

$rstin(BASE + 0xA);$ // 汎用入力ポートからの読み込み

表3-18A 【BASE+0xA】入力ポートの構成

ビット	各ビットの機能・意味	=1の時	=0の時
B7 ~ B1	未使用		
B0	汎用デジタル入力ビット0	High (開放)	Low (オン状態)

$out(BASE + 0xA, GenOut);$ // 汎用出力ポートへの出力

表3-18B 【BASE+0xA】出力ポートの構成

ビット	各ビットの機能・意味	=1の時	=0の時	リセット時
B7 ~ B1	未使用			
B0	汎用デジタル出力ビット0	High	Low	0

注1 電源投入またはハードウェアリセット直後の汎用デジタル出力は“0”ですが、ソフトウェアによるリセット操作ではクリアされません

注2 汎用デジタル出力の論理はボード上のショートプラグS-POLにより任意に設定する事ができます。出力レベルはTTLレベルです

3-19 割り込み制御

(本ポートについては、将来的に割り込み処理を行う前提で記述していますが、現状のドライバソフトウェアでは割り込み処理については未対応です)

out (BASE+0x4, imode); //割り込み要因のセットを行う

本ボードからパソコンへの割り込み要求を行う割り込み発生要因を制御します。複数の要因を許可するとオア動作となります。

表3-19A【BASE+0x4】出力ポートの構成

ビット	各ビットの機能・意味	=1の時	=0の時	リセット時
B7	外部割り込み信号の有効極性指定	↑ (+)	↓ (-)	
B6	ノットハーフフル遷移による割り込み	許可	禁止	0
B5	ノットフルによる割り込み	許可	禁止	0
B4	1ブロック出力開始 (SYNC) による割り込み	許可	禁止	0
B3	指定ブロック数出力終了による割り込み	許可	禁止	0
B2	トリガ発生による割り込み	許可	禁止	0
B1	外部割り込み信号による割り込み	許可	禁止	0
B0	クロックによる割り込み	許可	禁止	0

補足説明

B5：内部メモリの保持データ数が半分以下に遷移した条件で割り込みを制御する

B4：クロック同期更新モードで1ブロック分のDA出力開始により割り込みを制御する

B3：クロック同期更新モードで指定ブロック数分のDAデータ点数出力完了タイミングによる割り込み制御

B0：クロック同期更新モード時のDAデータ更新タイミングであるクロックによる割り込み制御

割り込み要求クリア

out (BASE+0x1, 0x0); //割り込み要求信号クリア

out (BASE+0x1, 0x1); //割り込み要求信号出力許可

本レジスタにソフトウェアから書き込みを行うことで本ボードからの割り込み要求を制御することができます。

表3-19B【BASE+0x1】出力ポートの構成

ビット	各ビットの機能・意味	=1の時	=0の時	リセット時
B7～ B1	未使用			
B0	PCIバスへの割り込み信号出力制御	出力許可	出力禁止 (クリア)	0

3-20 SYNC・クロック出力切り替え

out (BASE+0x9、Syn)

外部TTL出力信号をSYNC出力とするかクロック出力とするかの選択を行う。

クロック出力：クロック同期更新モードで使用される連続DAデータの更新タイミング信号（TTL立下り）

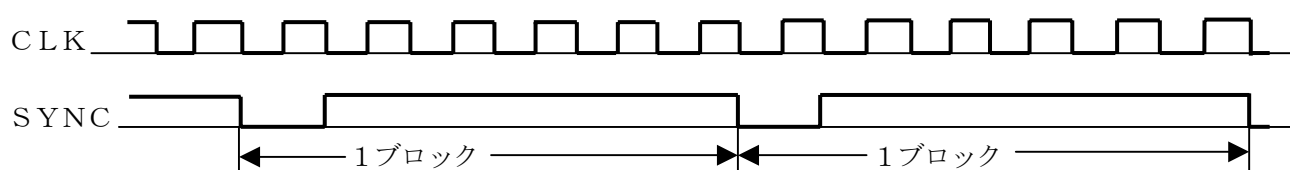
即ちクロック出力

SYNC出力：クロック同期更新モードのとき、1ブロックDAデータ群の先頭データ出力タイミング（立下り）を示す出力

表3-20【BASE+0x9】出力ポートの設定

ビット	各ビットの機能・意味	=1の時	=0の時	リセット時
B7～ B1	未使用			
B0	SYNC/クロック出力切換え	SYNC信号を出力	クロックを出力	0

図3-20 CLK/SYNCタイミング例



クロックのパルス幅：デューティは50%±1クロック源周期

SYNCのパルス幅：1クロック周期

両信号の有効エッジ：立下りエッジ

本例は、1ブロックデータ=6の場合

3-21 アナログ回路設定情報取得

inp (BASE+0xC)

本ボード内部のアナログ回路切り替え設定情報を取得する事ができるレジスタです。ビット0からビット3までの4ビットでアナログ回路設定情報を、ビット4からビット6の3ビットでFIFOメモリのハーフフルサイズを取得する事ができます。その詳細内容は、表3-20に示しています。

表3-21【BASE+0xC】入力ポートの構成

ビット	各ビットの機能・意味	=1の時	=0の時
B6～ B4	FIFOメモリハーフフルサイズ指定ビット	表3-21Bに詳細を示す	
B3	バイポーラ・ユニポーラ切り替え	ユニポーラ	バイポーラ
B2	基準電源内部・外部切り替え	内部基準電源	外部電源
B1	Aモード・Bモード切り替え	Bモード	Aモード
B0	出力バッファオンオフ切り替え	オフ	オン

表3-21B FIFO メモリハーフフルサイズ指定ビット構成

B 6	B 5	B 4	ハーフフルサイズ
0	0	0	5 1 2 語
0	0	1	1 0 2 4 語
0	1	0	2 0 4 8 語
0	1	1	4 0 9 6 語
1	0	0	8 1 9 2 語
1	0	1	1 6 3 8 4 語
1	1	0	3 2 7 6 8 語
1	1	1	6 5 5 3 6 語

3-22 アナログ回路情報設定

o u t (BASE+0x B, (BYTE) Mode)

本ボード内部のアナログ回路切り替え情報を設定できるレジスタです。ビット0からビット3までの4ビットでアナログ回路情報を、ビット4からビット6の3ビットでFIFOメモリのハーフフルサイズを夫々設定することができます。その詳細内容は、表3-21に示しています。

表3-22【BASE+0x B】出力ポートの構成

ビット	各ビットの機能・意味	= 1 の時	= 0 の時	リセット時
B 6 ~ B 4	FIFOメモリハーフフルサイズ指定ビット	表3-22B に詳細を示す		
B 3	バイポーラ・ユニポーラ切り替え	ユニポーラ	バイポーラ	0
B 2	基準電源内部・外部切り替え	内部基準電源	外部電源	0
B 1	Aモード・Bモード切り替え	Bモード	Aモード	0
B 0	出力バッファオンオフ切り替え	オフ	オン	0

表3-22B FIFO メモリハーフフルサイズ指定ビット構成

B 6	B 5	B 4	ハーフフルサイズ
0	0	0	5 1 2 語
0	0	1	1 0 2 4 語
0	1	0	2 0 4 8 語
0	1	1	4 0 9 6 語
1	0	0	8 1 9 2 語
1	0	1	1 6 3 8 4 語
1	1	0	3 2 7 6 8 語
1	1	1	6 5 5 3 6 語

但し、実際のレジスタ値変更の際には、予めハードウェア上エンコードされているため、細かなテクニックが必要です。具体的には

- ・まずオフセットBのレジスタに対して0を出力します。
- ・次にオフセットCのレジスタから値を読み出す事でハードウェアエンコード情報を取得します。
- ・本項目の内容に従い、必要なビットマップコーディングを行います。
- ・最後に作成したビットマップ情報と、第2項で読み出したハードウェアエンコード情報との間で排他論理和を計算し、その結果をオフセットBのレジスタへ出力します。この際、専用ハンドラーでは、変更されたハーフフルサイズビットに従い、内部で保持しているハーフフル設定値を更新します。

3-23 搭載メモリ量の取得

このポートから読出しを行うことで、本ボードに搭載されているメモリサイズを判定することができます。現状取り得る値は0（8M語）、1（16M語）、2（32M語）及び4（64M語）の4種類ですが、将来的にはそれ以上の容量への対応も可能です。

表3-23【BASE+0x9】入力ポートの構成

ビット	各ビットの機能・意味	
B7～ B0	搭載メモリサイズ	0x00：8M語 0x01：16M語 0x02：32M語 0x04：64M語

3-24 メモリ使用量の取得

このポートから連続して4回データを読み込む事で、内部メモリデータ使用量を任意のタイミングで確認することができます。

表3-24【BASE+0x8】入力ポートの構成

ビット	読込み1回目	読込み2回目	読込み3回目	読込み4回目
B7～ B0	カウンタ値 D7～0	カウンタ値 D15～8	カウンタ値 D23～16	カウンタ値 D31～24

本ポートについては、表3-24に示すように必ず4回のデータ読出しを行うことでカウンタ値を取得できるように設計されていますので、決められた読込みシーケンスを守る事が重要です。

つまり

```
count = 0
```

```
count = inp (BASE+0x8)
```

```
count |= (inp (BASE+0x8) << 8)
```

```
count |= (inp (BASE+0x8) << 16)
```

```
count |= (inp (BASE+0x8) << 24)
```

という一連の手続きによりメモリ使用量が求められるということになります。

3-25 複数ボードの同期運転

複数の本ボードを同一クロックで同期運転することができます。この場合、トリガを検出し動作を開始するボードをマスターボード、その他のボードをスレーブボードとします。各スレーブボードはマスターボードからのクロックを受けて同期を取りますが、サンプリングタイミングに最大100nSの遅れが生じます。

全体の接続、操作等は以下のとおりとします。

ボード上の設定

各ボードのSW-BNスイッチについては、マスターボードを“0”、スレーブボードを非“0”とし、更にスレーブボード間でその値がユニークになるような設定を行う必要があります。通常スレーブボードは“1”、“2”等連続した非“0”とする事が望ましいと考えられます。

ボード間の接続

ボード間の接続については、マスターボードのクロック出力を各スレーブボードのクロック入力に接続するだけです。マスターボードについては、外部クロック、外部トリガを使用しても問題ありません。

ソフトウェア

各スレーブボードのクロック源は外部、分周比は1／1と設定します。

各スレーブボードの外部トリガは禁止状態とします。

マスターボードのクロック、トリガについては単一ボードの場合と全く同様に行え特に制限ありません。

スレーブボードのトリガ条件については、ビット7は無条件にセット、ビット6からビット4は全てクリア、ビット3からビット0については、マスターボードと全く同じ設定とします。(そのようにしないとスレーブ側が動作しない事があります。)

スタート操作は、最初に各スレーブボードを（ソフトトリガで）スタートさせ、最後にマスターボードのスタートを任意の条件で開始するというシーケンスで行います。

3-26 外部機器との連携動作について

クロックの入出力機能を利用すると、外部イベントにリアルタイムで応答する事や、外部機器に動作・測定等の正確なタイミングを伝える事ができます。

応用例：任意波形出力及びADサンプリング

ボード間の接続： 本ボードのクロック出力をADボードのクロック入力に接続する

ADボード側： 外部クロック動作に設定しソフトトリガで外部クロック待ち状態としておく

DAボード側： 出力波形データを用意する。任意のクロック源及びトリガにより出力を開始する

注意点： DA出力がクロック同期・サイクルモードなら以後はADボードからデータを読み込む事に専念できる。クロック同期非サイクルモードの時は、DAデータの追加転送とADボードからのデータ読みを遅れることなく調和させる必要がある。

第4章 ソフトウェア

4-0 ハードウェアとソフトウェアの連携

前章までで解説したハードウェアと、ユーザーアプリケーションとを接続する役目を果たしているのが、本章で説明するデバイスドライバと呼ばれるソフトウェアになります。その動作はかなり複雑ですが、視覚的に簡略化した相互の関係を図4-0に示します。

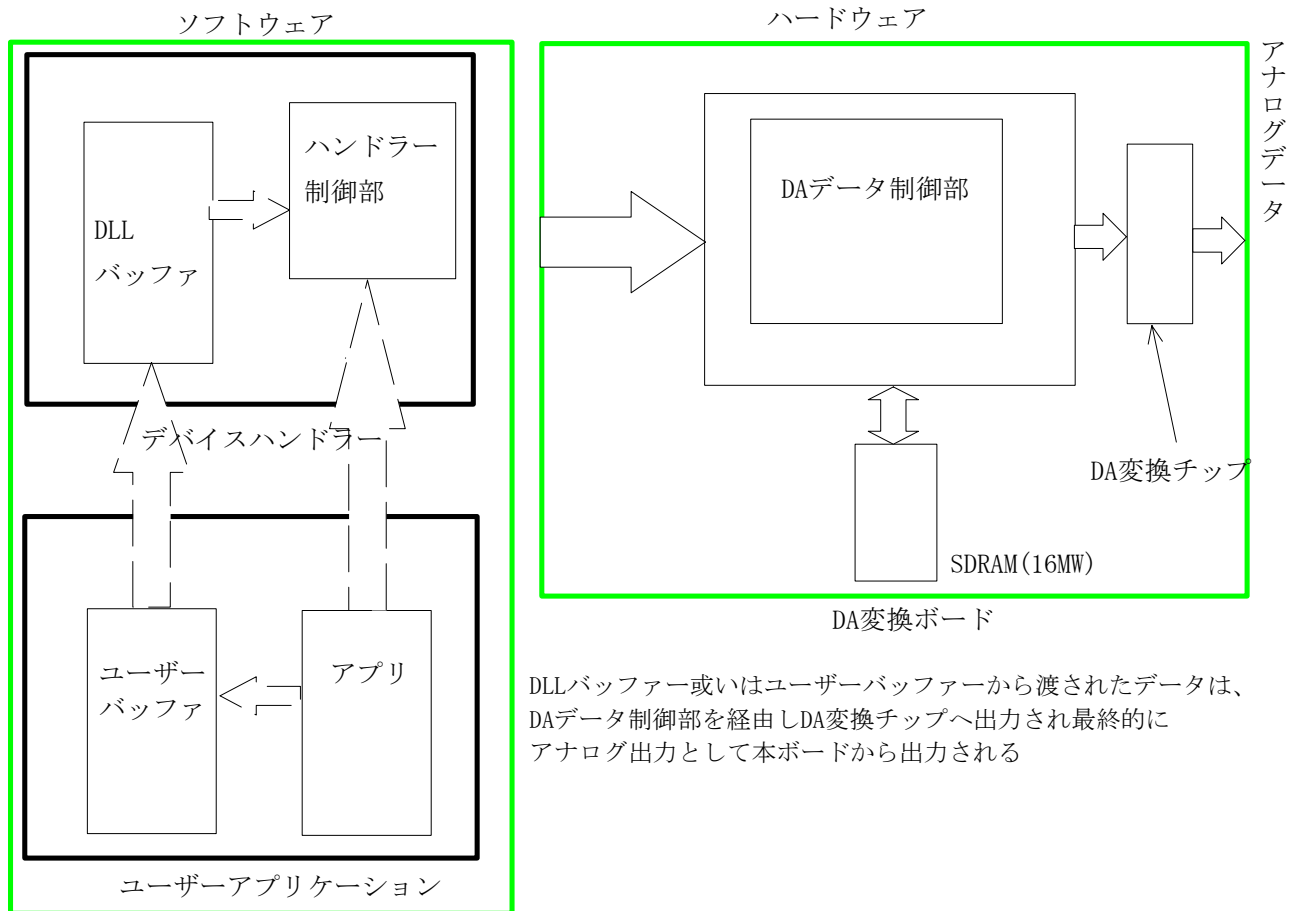


図4-0 ハードウェア v s ユーザーアプリケーション関連図

DA変換ボード内の全てのリソースは、ソフトウェアサイドから見ると入出力レジスタにマッピングされています。MSDOSの時代であれば、ユーザーアプリケーションサイドから直接入出力命令を発行する事により、ハードウェアの制御ができましたが、現在ではそのような操作はセキュリティ上の問題から禁止されており、これらレジスタの操作は、デバイスハンドラーというシステムサイドのソフトウェアを介在させて行うことになっています。

本章の以下の部分では、このデバイスドライバを使用したハードウェア制御の方法を、順序だてて説明してゆく事になります。

尚、本デバイスドライバはマイクロソフト製ビジュアルスタジオ 2008 を使用して製作されているため **ビジュアルスタジオ 2008 用ランタイムライブラリが同時にインストール**されます。

4-1 本製品用ソフトウェアの種類

本製品用のソフトウェアは、表4-1に示す3種類から構成されています。即ち、WindowsXP(32Bit)対応のデバイスドライバー、Windows7(32Bit)対応のデバイスドライバー及びWindows7(64Bit)対応のデバイスドライバーの3種類です。また、夫々のOS用に専用ハンドラー並びに汎用IOハンドラーが用意されています。

表4-1 ドライバー構成表

OS及びビット幅		Windows7 32Bit	Windows7 64Bit	WindowsXP 32Bit
パッケージ名称		Redist0112_x86.zip	Redist0112_x64.zip	Redist0112_xp.zip
専用ドライバー	ドライバー(DLL/Lib)	FCDA0112PCI_X86	FCDA0112PCI_X64	FCDA0112PCI_X86
	関数宣言	FCDA0112PCI.H	FCDA0112PCI.H	FCDA0112PCI.H
汎用ドライバー	ドライバー(DLL/Lib)	PORT7_X86	PORT7_X64	PORT7_X86
	関数宣言	PORT7.H	PORT7.H	PORT7.H

WindowsXP(32Bit)対応のドライバーは、アプリケーション移植時の利便性を考慮し用意されているものです。このドライバーを経由する事により、ドライバー更新→OS変更というステップを2段階に分割して実行できるように考慮されています。ドライバーそのものはWindows7(32Bit)版と同じものになっていますが、インストール方法の違いによりパッケージの構成がWindowsXP対応の形になっています。

またドライバー内の関数セット名称の内、同一機能のものについてはマイクロサイエンスのドライバーと同一とし移植時の利便性を考慮しています。

4-2 ドライバーのインストール

4-2-1 WindowsXP ドライバーインストール及びアンインストール

WindowsXP 対応ドライバーを添付CDROMよりコピーし、その内容を全て解凍します。解凍場所はCドライブのルートを指定して解凍します。解凍結果は以下のようになります。

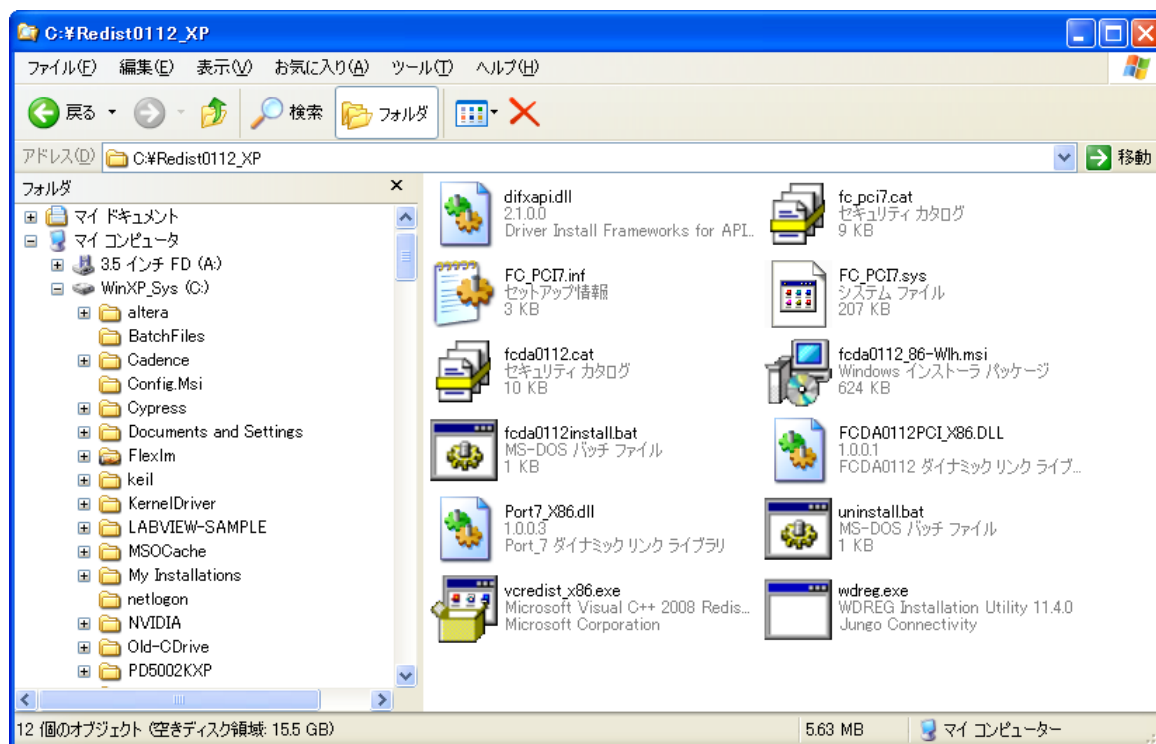


図4-2-1 A

デバイスインストールを開始する前に、旧ドライバーがインストールされていた場合、これを削除しなければなりません。そのためには、デバイスマネージャを開き、マイクロサイエンスドライバーが登録されているかどうかを確認します。因みに、下の図は、参考例としてHDA-770PCIがインストールされている状態で説

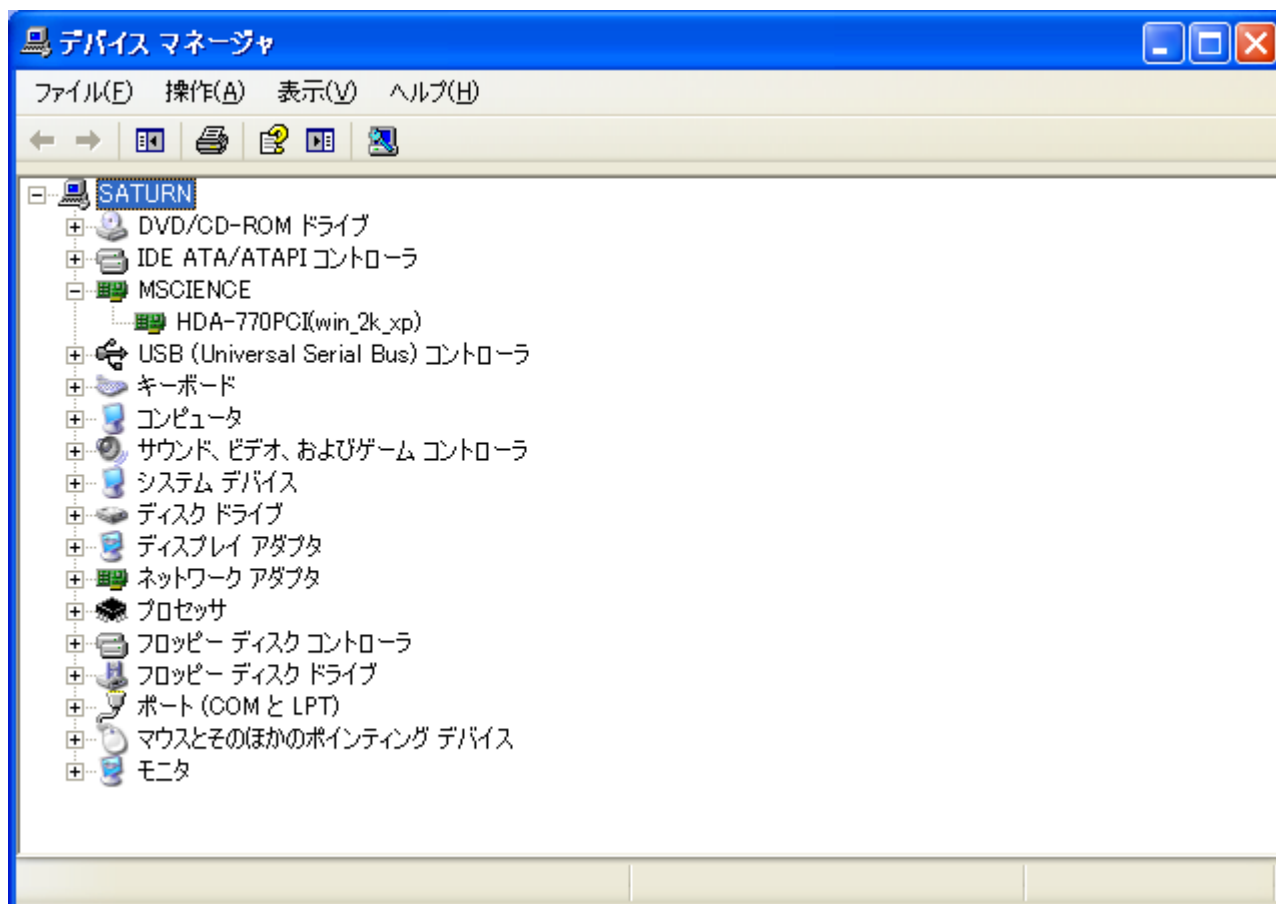


図 4-2-1 B

上図に示すように MSCIENCE というエントリが存在すれば、旧ドライバーがインストールされています。
ドライバー削除は、

- 1 まず Windows¥Inf フォルダの中から Oem*.Inf ファイルを検索します。検索条件としては、文字列として “MSCIENCE” が含まれているものを条件とします。検索されたファイルをテキストエディターでマイクロサイエンスのものである事を確認し、Oemxx.Inf (xx は上記の条件に合致した番号です) と Oemxx.Pnf を削除します。
- 2 次に Windows¥System32 フォルダの中から、DMS_PCI.Sys を探し同じように削除します。
- 3 最後に、図 4-2-1 B で示すマイクロサイエンス製ドライバー（例では HDA-770PCI です）を右クリックし削除します。

ここまできた段階で一旦パソコンの電源を落とし、ボードを取り外します。

再度パソコンの電源を投入し、図 4-2-1 A で示されるフォルダの中から “Fcda0112PciInstall.Bat” ファイルを右クリックし “開く” (実行する) を選択します。すると、まず図 4-2-1 C で示されるウィンドウが開きます。(次ページ)

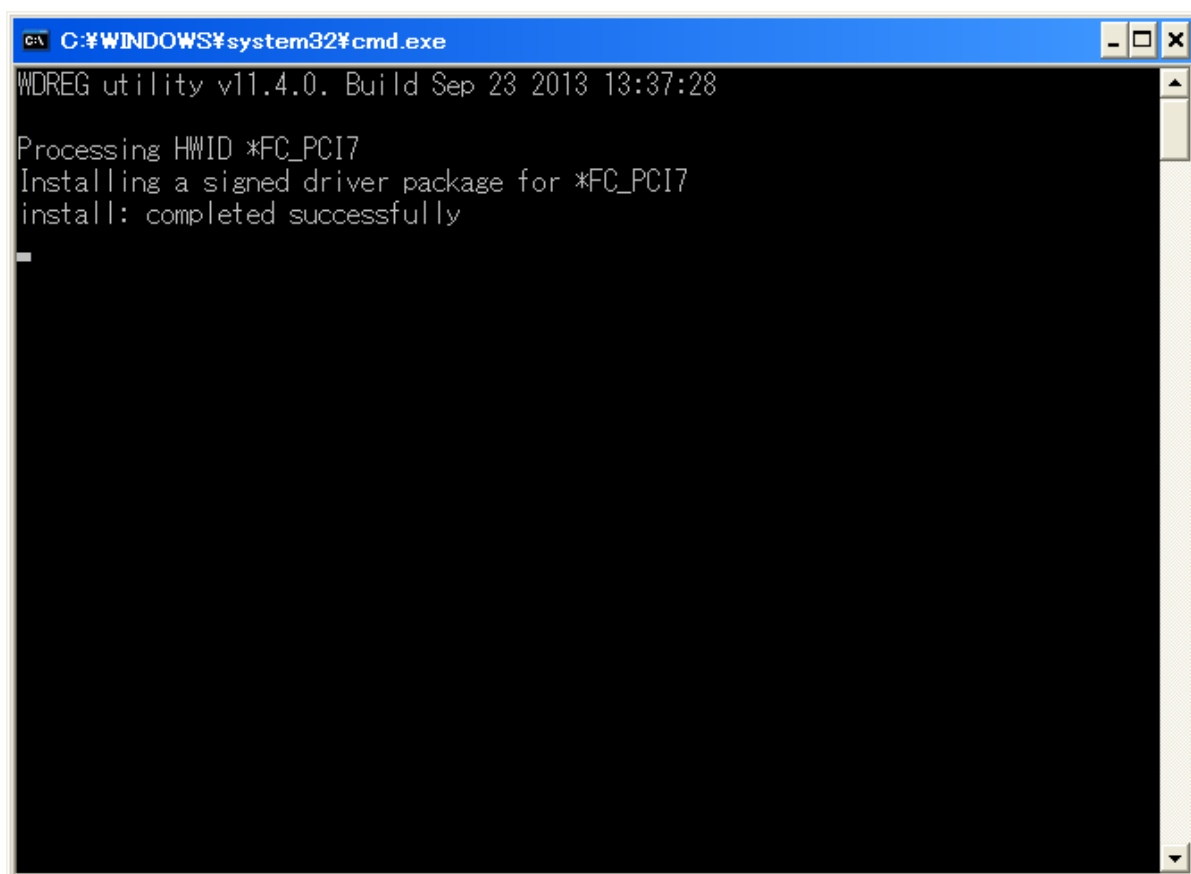
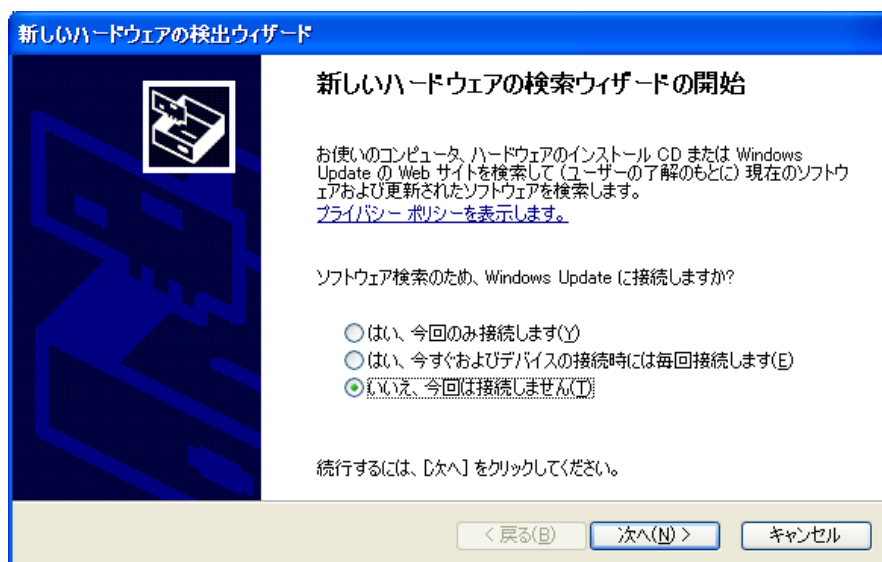
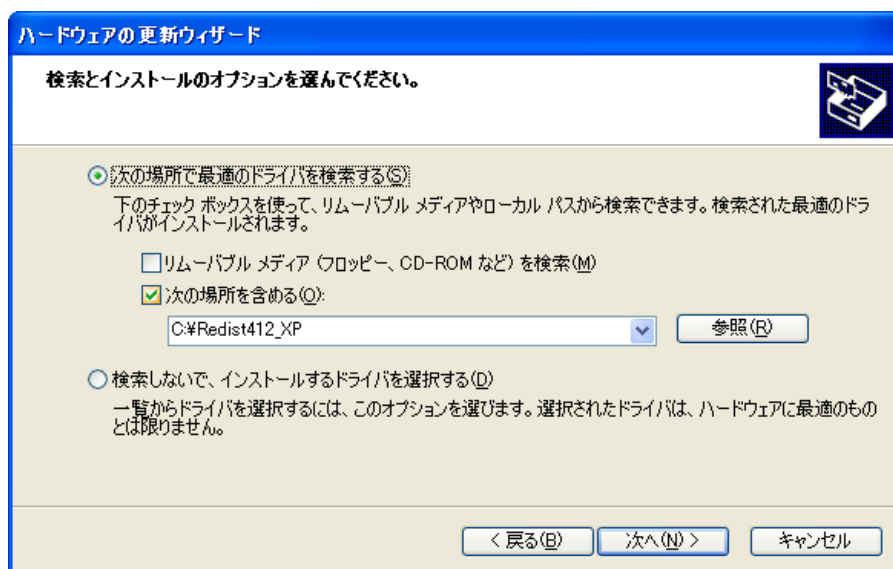
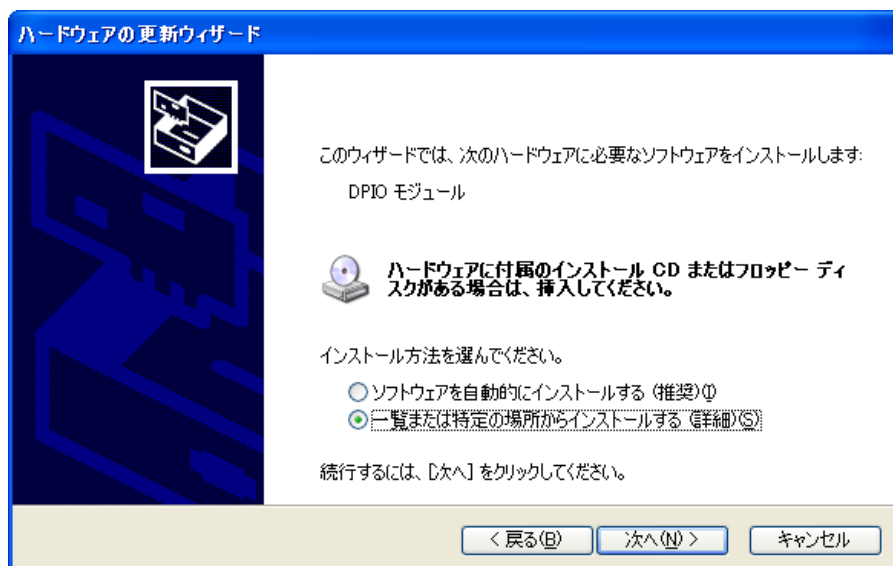


図 4-2-1 C

そして暫くすると、インストール作業が終了します。そこで再度パソコンの電源を遮断し FCDA0112PCI ボードを再度挿入し電源を投入します。システムが立ち上がるとハードウェアウィザードが立ち上がりますから、



WindowsUpdate への接続をせず、“次へ”ボタンをクリックします。そして次の画面（次頁）で、次のハードウェアとして“FCDA0112PCI”が表示された場合は、インストール方法として“ソフトウェアを自動的にインストールする”を選択し“次へ”ボタンをクリックします。すると必要な DLL ファイル群がシステムフォルダにコピーされインストール作業は終了します。また次頁のようにハードウェアとして“DPIO モジュール”が表示された場合は”一覧または特定の場所からインストールする”を選択し“次へ”ボタン押下後の画面でインストール先として“C:\¥Redist0112_XP”を選択しインストールを行います。



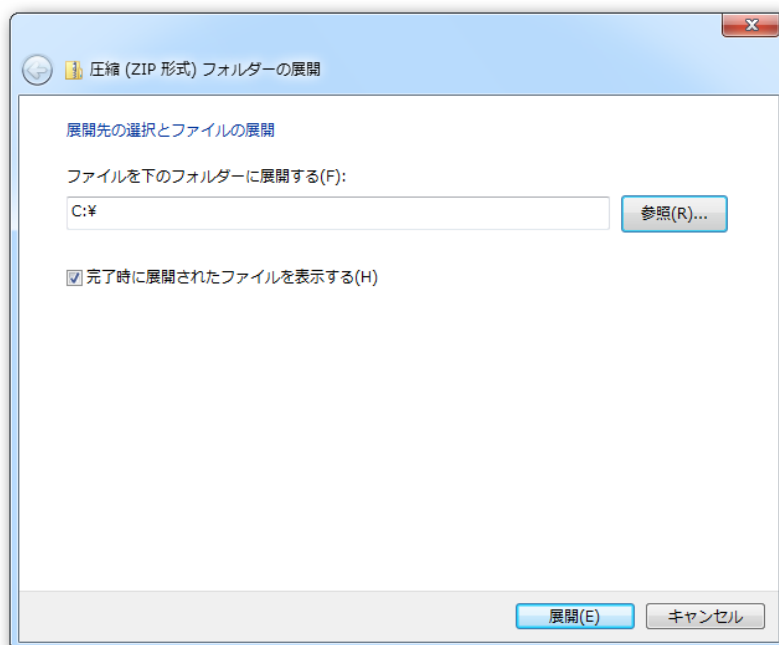
また、ドライバーを削除する場合は、図 4-2-1 A で “Uninstall.Bat” ファイルを実行することで自動的にドライバーが削除されます。

4-2-2 Windows 7 (X86 : 32 ビット) へのインストール

本製品用のソフトウェアは添付 CDROM に収納されています。その中から、実際に御使用になるOSとマッチしたドライバーセットをパソコンにコピー、展開後、デバイスドライバーをインストールしていただきます。

実際のインストール方法は以下の通りです。

- ・ 本ボードをパソコンにセットアップする前に、添付CDROMよりご使用のOS用デバイスドライバーキットをパソコンにコピーし解凍します。解凍場所は、Cドライブのルートとします。（本ドライバーキットには、デバイスハンドラーが含まれています。）



このウィンドー内の“展開” ボタンをクリックするとドライバーセットがCドライブ内に展開されます。内容は図4-2-2 Aを参照してください。

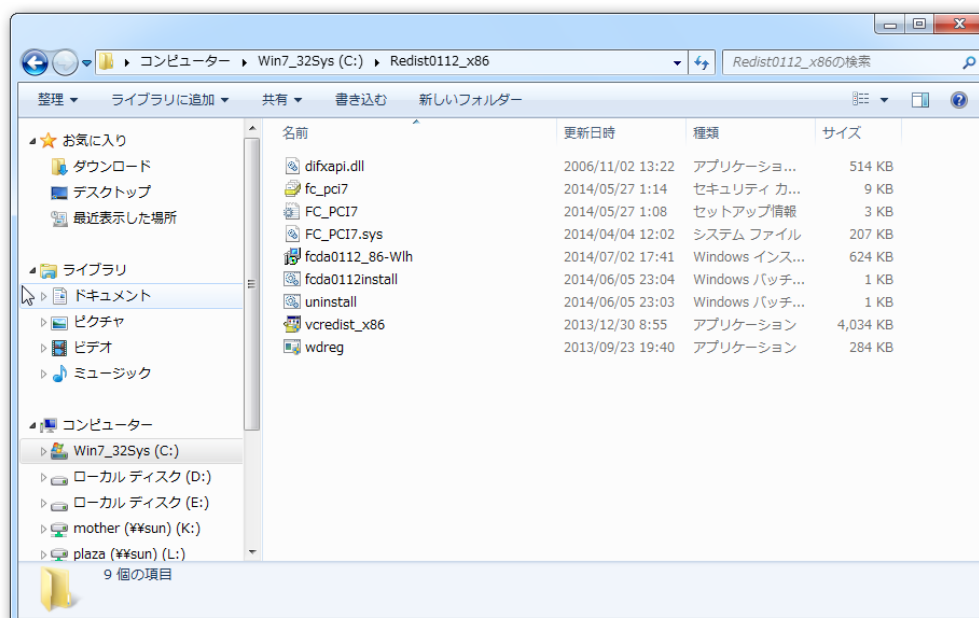
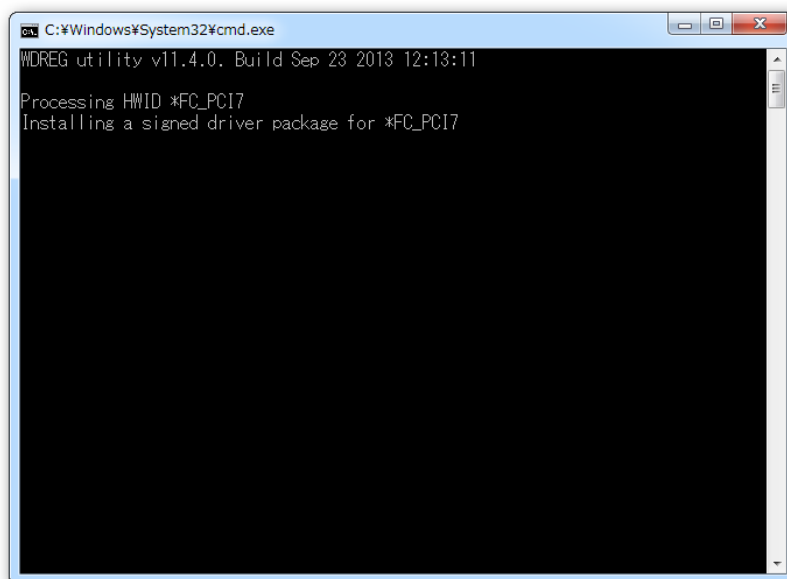


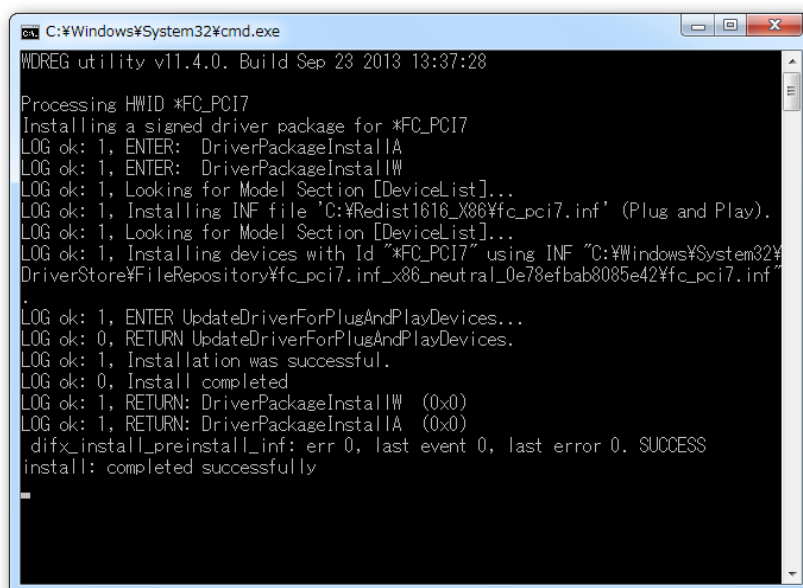
図 4-2-2 A

この中で“Fcda0112Install.Bat” がデバイスドライバーインストーラーです。また、この時点で、本ボードは、まだパソコンには取り付けないで下さい。この順序が守られない場合、正常なインストールは保証されません。

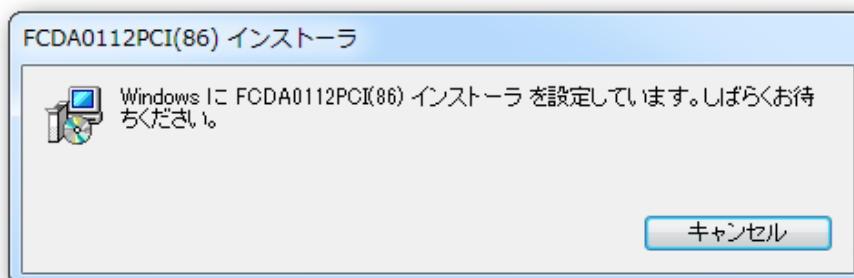
このデバイスドライバインストーラーを**管理者として実行**（右クリックメニューから選択）すると次のウィンドーが開きます。（ただ実行とすると、ウィンドーが瞬間的に終了しデバイスドライバのインストールはできません）最初に次のウィンドーが開きます。



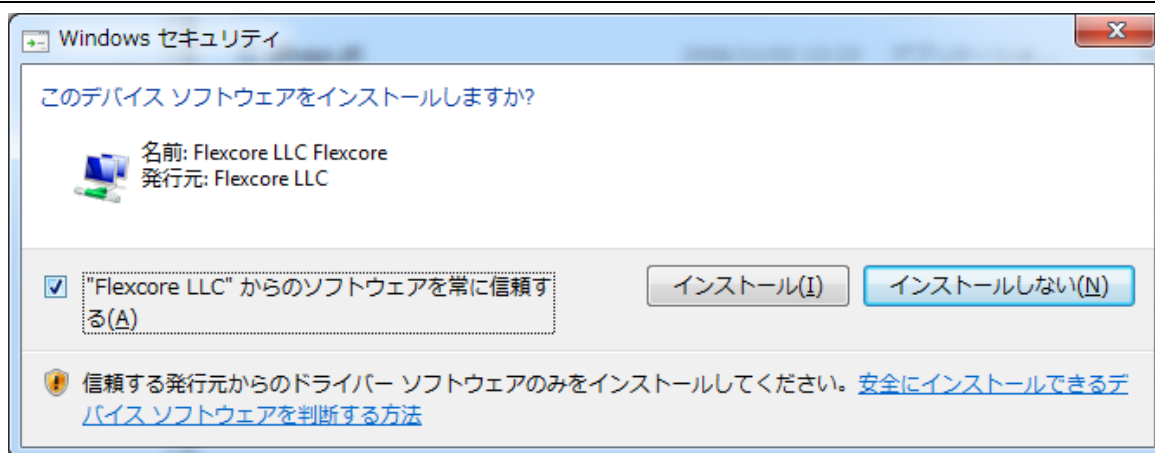
次いで次のウィンドーが開きます。



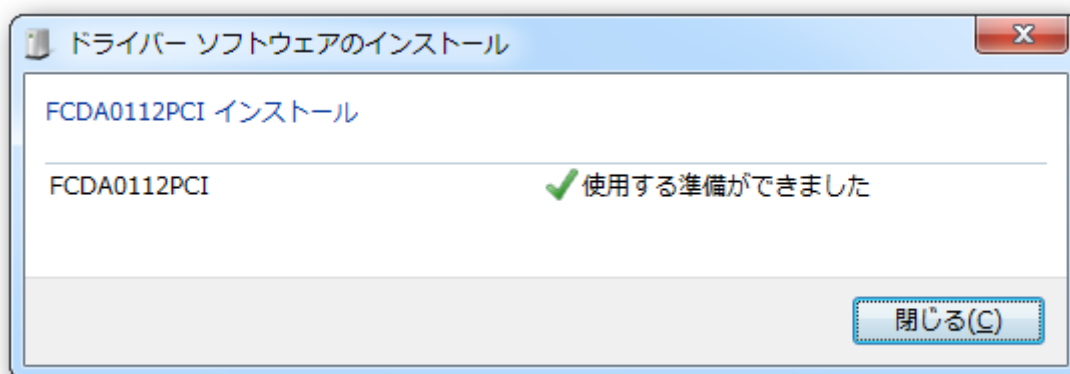
更に暫く待っていると、次のウィンドーが開きます。



更に待っていると次のウィンドーが開きます。（次ページ）



ここでは、“Flexcore LLC”からのソフトウェアを常に信頼するという項目にチェックを入れ“インストール”ボタンをクリックします。そして暫く待っているとインストール作業が終了します。この後、一旦パソコンの電源を落とし、パソコン内部のPCIスロット或いはPCI拡張ユニットに本ボードを挿入し再度パソコンの電源を投入します。OSが立ち上がるとハードウェアの検出が行われ、少し待っていると画面下部のタスクバーの所に、デバイスドライバーが正常にインストールされたというメッセージが表示されます。



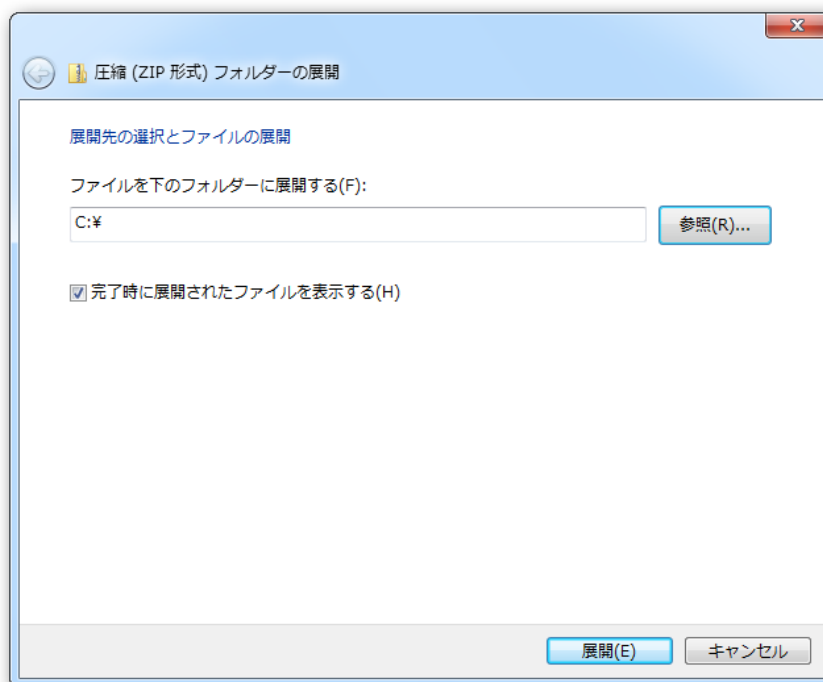
これで本ユニットのインストールは完了しました。

4-2-3 Windows 7 (X64 : 64 ビット) へのインストール

本製品用のソフトウェアは添付CDROMに収納されています。その中から、実際に御使用になるOSとマッチしたドライバーセットをパソコン内にコピーし、本ボードを取り付ける前にデバイスドライバーをインストールしていただきます。

実際のインストール方法は以下の通りです。

- 本ボードをパソコンにセットアップする前に、添付CDROMよりご使用のOS用デバイスドライバーキットをパソコン内にコピーし、その中に解凍します。解凍場所は、Cドライブのルートとします。（本ドライバーキットには、デバイスハンドラーが含まれています。）



ここで“展開” ボタンをクリックするとドライバーセットがCドライブ内に展開されます。内容は図4-2-3Aを参照してください。

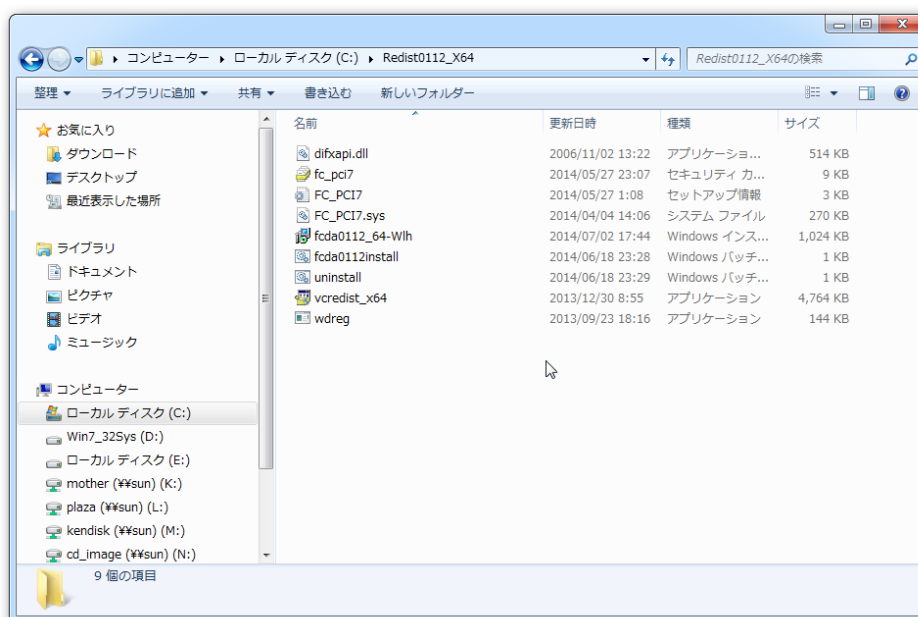
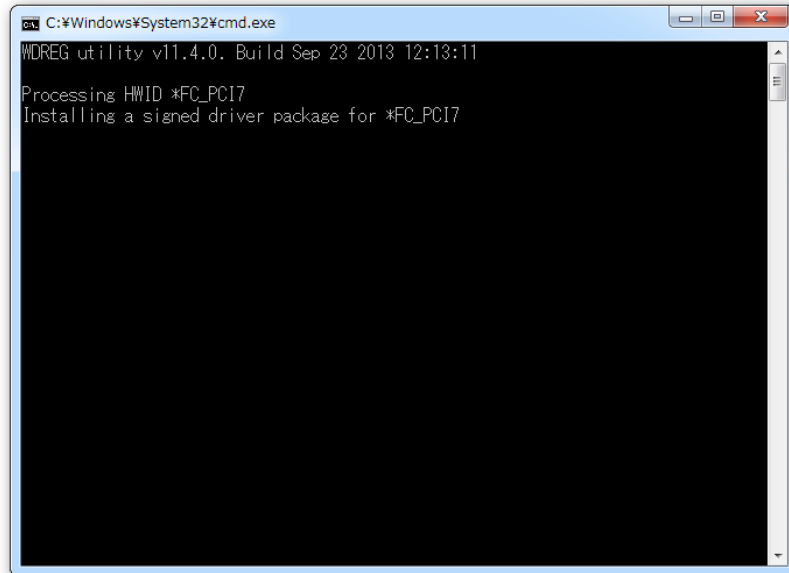


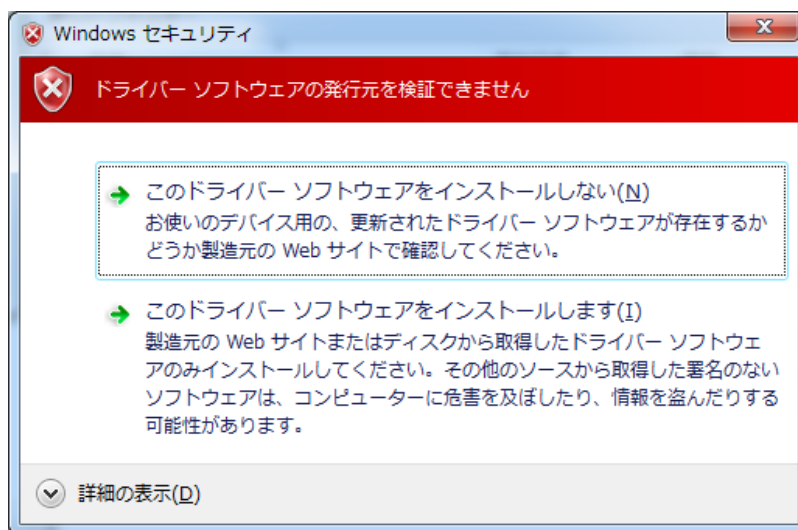
図4-2-3A

この中で“Fcda0112Install.Bat”がデバイスドライバインストーラーです。また、この時点で、本ボードは、まだパソコンには取り付けないで下さい。この順序が守られない場合、正常なインストールは保証されません。このデバイスドライバインストーラーを**管理者として**実行（右クリックメニューから選択）すると次のウィンドーが開きます。（ただ実行とすると、ウィンドーが瞬間的に終了しデバイスドライバのインストールはできません）

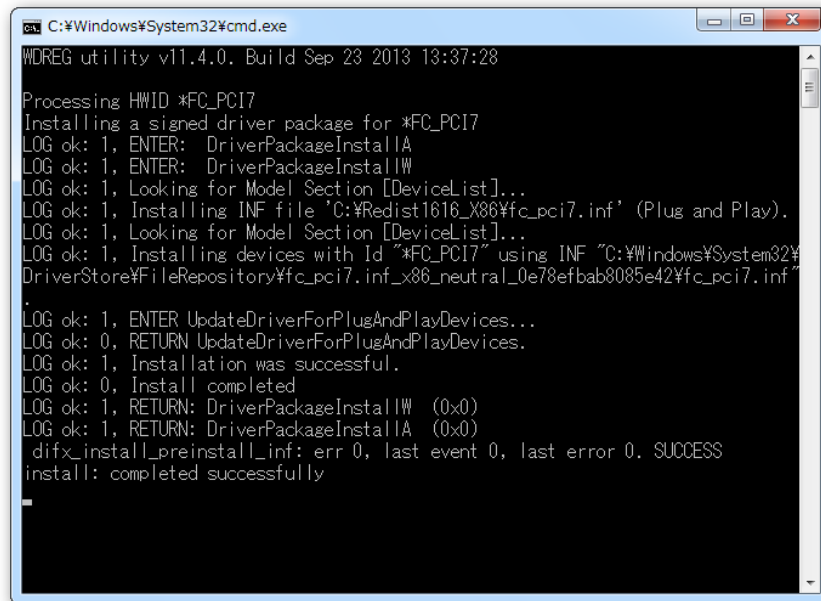
最初に次のウィンドーが開きます。



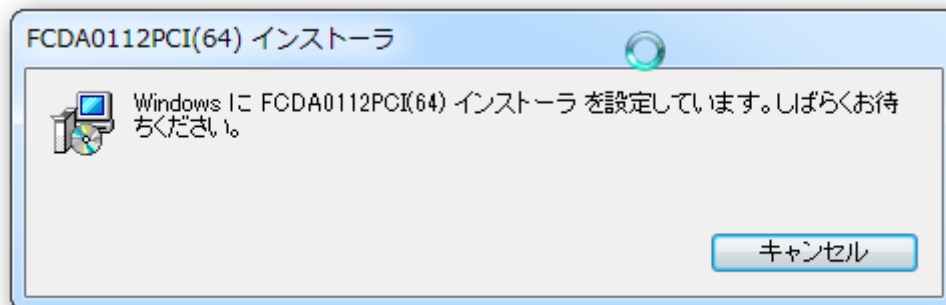
次いでセキュリティ警告を示すウィンドーが開きます。



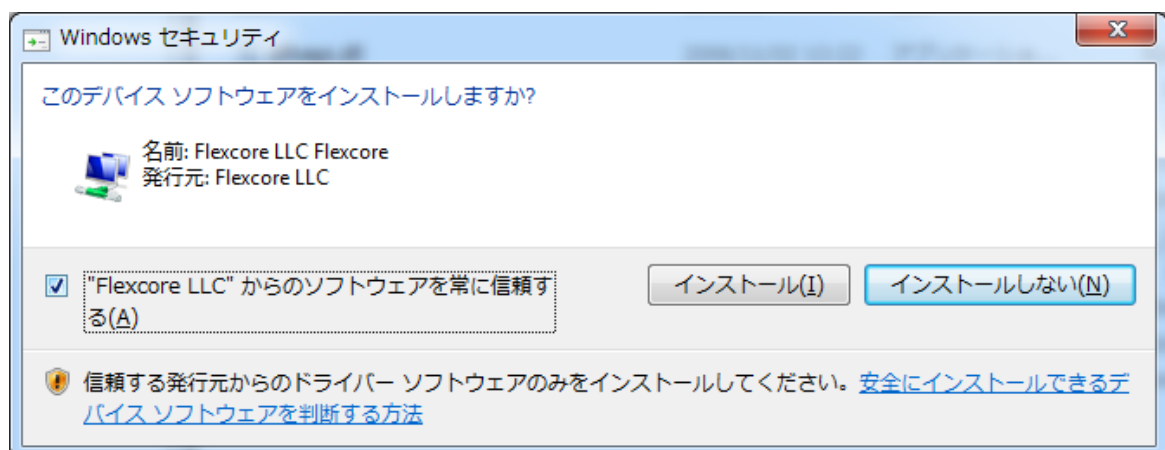
ここでは、“このドライバーソフトウェアをインストールします”側のボタンをクリックして下さい。すると次のウィンドーが表示されます。（次ページ）



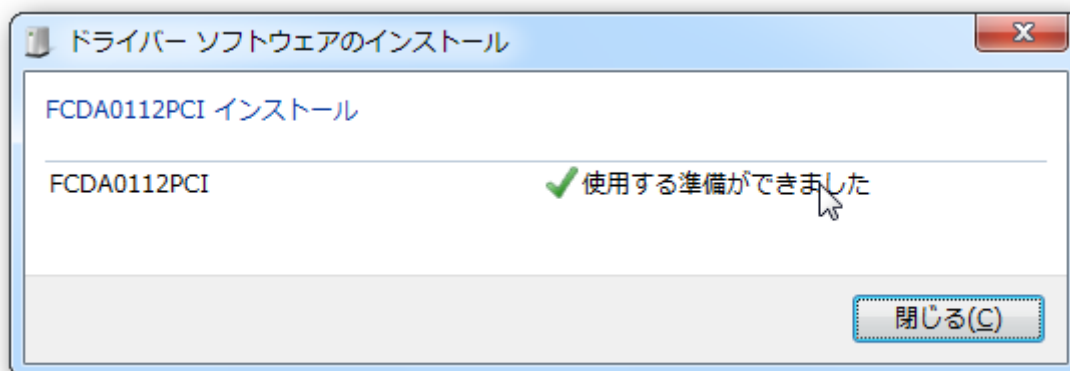
更に暫く待っていると、次のウィンドーが開きます。



更に待っていると次のウィンドーが開きます。



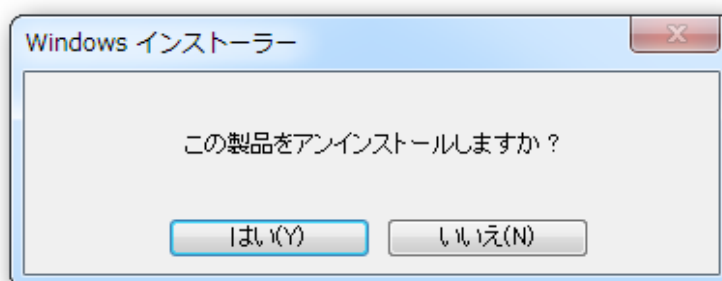
ここでは、“Flexcore LLC”からのソフトウェアを常に信頼するという項目にチェックを入れ“インストール”ボタンをクリックします。そして暫く待っているとインストール作業が終了します。この後、一旦パソコンの電源を落とし、パソコン内部のPCIスロット或いはPCI拡張ユニットに本ボードを挿入し再度パソコンの電源を投入します。OSが立ち上がるとハードウェアの検出が行われ、少し待っていると画面下部のタスクバーの所に、デバイスドライバーが正常にインストールされたというメッセージが表示されます。



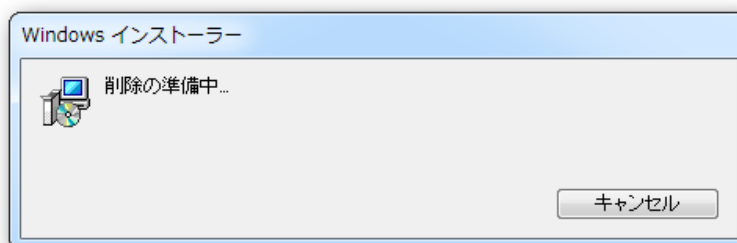
これで本ユニットのインストールは完了しました。

4-3 Windows 7 ドライバーのアンインストール

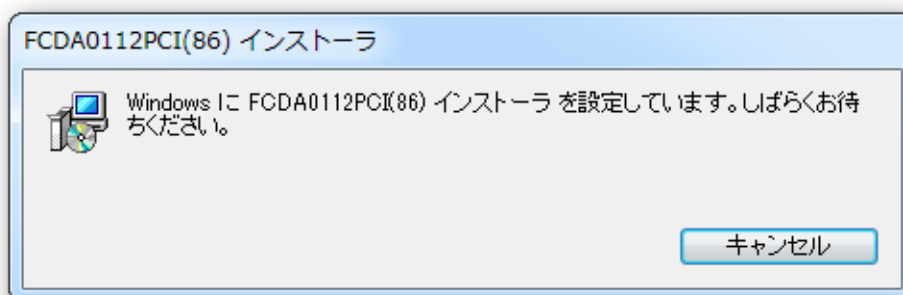
この方式でインストールしたドライバーはドライバー更新時などにドライバーをアンインストールする際にも簡単に行う事ができます。まず、4-2-1 項で説明したデバイスドライバー解凍フォルダへ移動し、この中にある “Uninstall.Bat” ファイルを実行します。（ここでも **管理者として** 実行する必要があります。）



というメッセージが現れますので “はい” をクリックします。すると

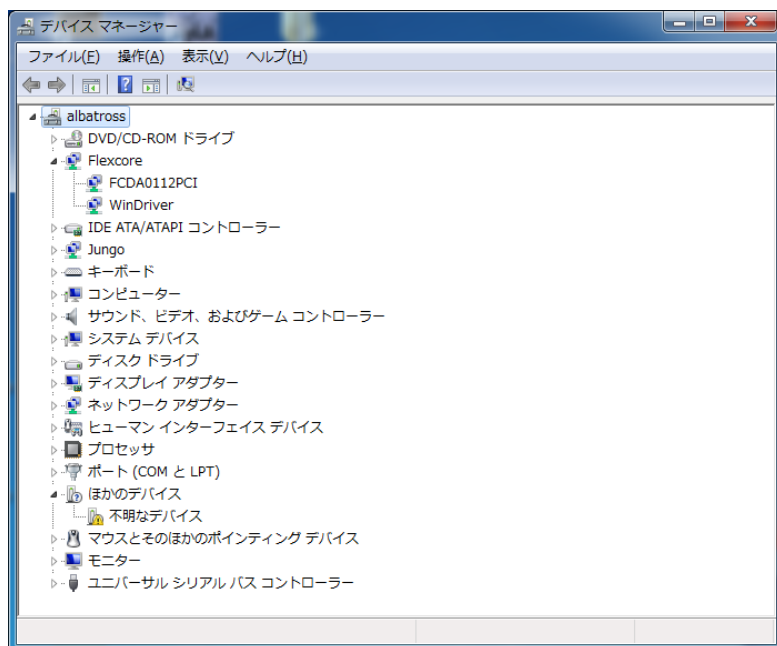


というウィンドーに変わり、更に待っていると



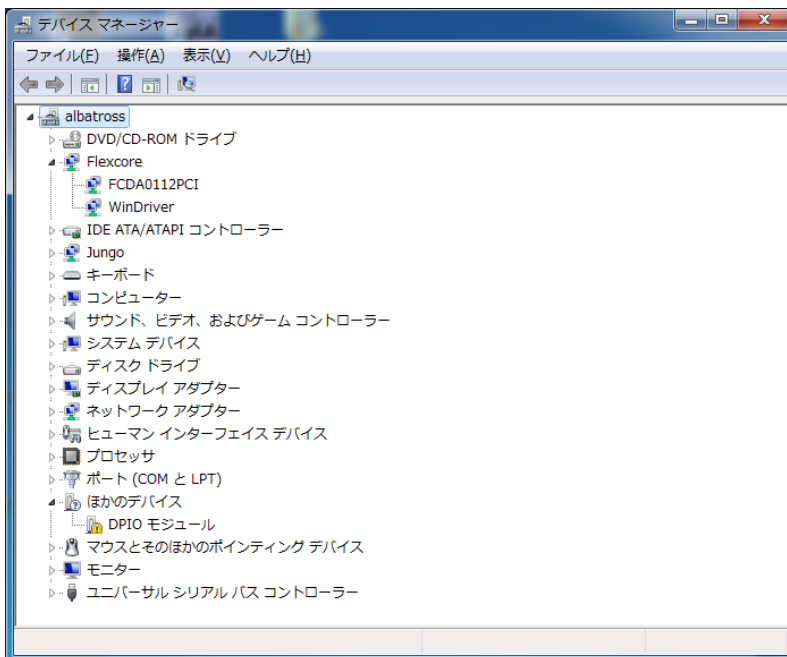
というウィンドーに変わり、暫く待っているとデバイスドライバーが削除されます。

この段階で、スタート>コントロールパネルと進み、デバイスマネージャを開くと、下図に示すように不明なデバイスが表示されています。(同図下から4行目)
他方、上から4行目にはアンインストールされていないボード情報も存在しています。



そこで、この“不明なデバイス”を右クリックし、表示されるドロップダウンリストから“デバイスの削除”を選び実行します。その結果、図4-3-2に示すように“不明なデバイス”がドライバー未インストール時と同様、“ほかのデバイス”中に DPIO モジュールとして認識されるようになります。(この表示を得るためには、トップエントリー：同図では albatross と表示されているエントリーを右クリックし“ハードウェア更新のスキキャン”を実行する必要があります。)

この状態に到達すると、パソコンの電源を遮断し、本ボードを取り外す事ができます。特にバージョンの異なるドライバーに変更する際などには、同図に示されるように、本ボードが DPIO モジュールという状態になっている必要があります。



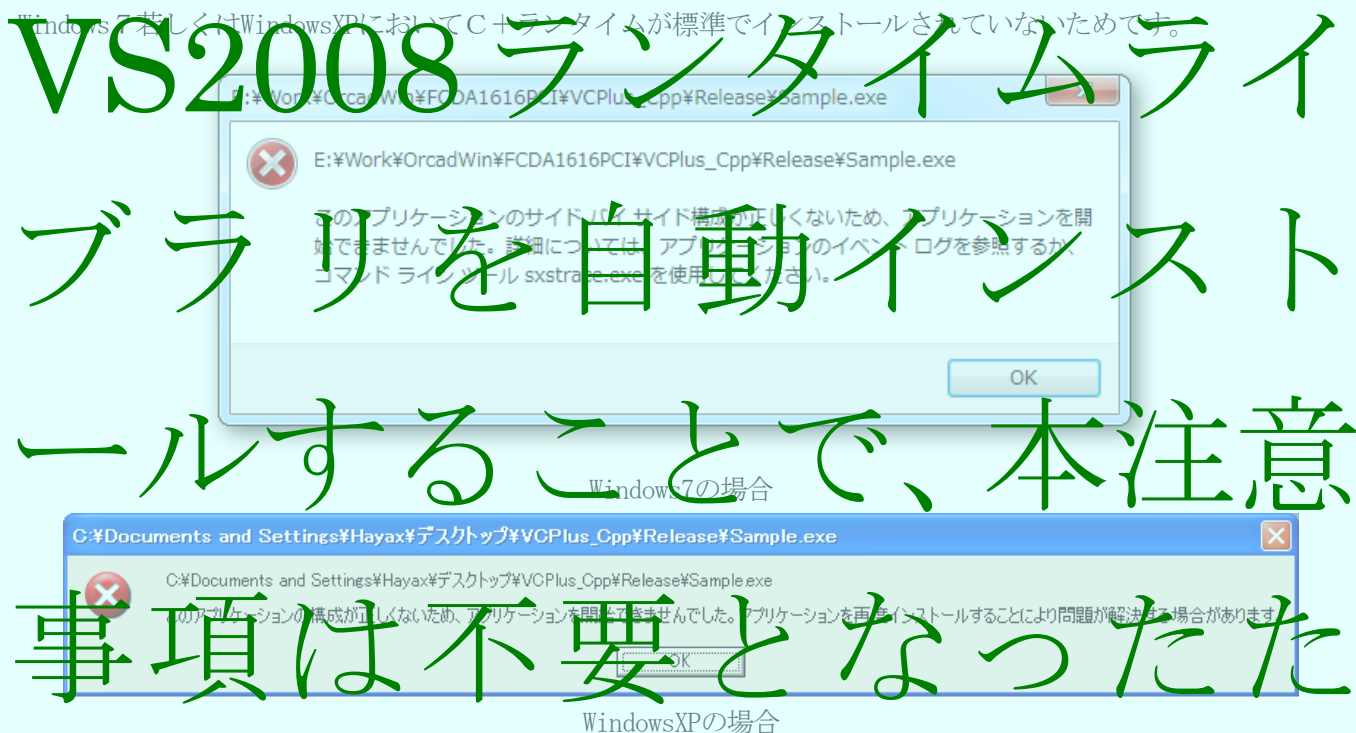
この状態で、電源遮断後、本ボードをシステムから取り除くと、4-2項 ドライバーインストール直前の状態に復帰している事になります。

4-4 サンプルプログラムについて

本製品のサンプルプログラムは、大きく2種類から構成されています。第一のプログラムは、汎用入出力関数を利用したボードの動作確認プログラムであり、(内部レジスタ動作確認及び簡単なDAデータ出力処理) 第二のプログラムは専用ハンドラーを使用したC++による自動サンプリング(Dll バッファ使用)、マニュアルサンプリング(ユーザーバッファ使用)及びループ処理プログラムの3構成から構成されています。また、アナログ回路設定変更などの実行例については、基本的に自動サンプリングサンプルの中で記述しています。また、X86(32ビット)及びX64(64ビット)の実行ファイルについてはそれぞれのサンプルプログラム内でディレクトリ構造によって住み分けを行っています。詳細については、添付CDROM内のパッケージ確認をお願いします。

~~4-5 サンプルプログラム、動作確認プログラムについて(補足)~~

サンプルプログラムを実際に動作させると、以下のようなメッセージが出て動作しない事があります。これは、Windows7若しくはWindowsXPにおいてC++ランタイムが標準でインストールされていないためです。



弊社で作成した動作確認プログラムはMSVC+2008でコンパイルされておりますので、専用のランタイムをインストールする事により、この状況を回避する事ができます。必要となるランタイムは、ドライバーパッケージの中に同梱しておりますのでこちらを実行することでランタイムのインストールができます。ファイル名称は、vc_redist_x86.exe (X86環境) 又はvc_redist_x64.exe (X64環境) になります。

若しくは、(まだC++環境をインストールされていない場合) 当該パソコンにMSVC+2008等の開発環境をインストールする事で対処する事ができます。(MSVC+2008以外の開発環境をインストールされる場合、実行ファイルについては、その環境で再構築する必要があります。)

め削除

4-6 関数仕様・エラーコード

本ボードの関数セットは、表4-4に示すように、マイクロサイエンス製ドライバー内の関数名と等価な関数及び名称が全く異なる関数群から構成されています。これは、アプリケーション移植に際して意味のない名称修正等を避け、移植ミスをできるだけ減らすためです。名称に互換性がない関数は、本ボードでハードウェア上追加された機能を使用する関数若しくは元々の関数セットには存在しなかった関数に限られていますので、アプリケーション上これらの機能を使用する必要がない場合、アプリケーション側の修正箇所は、ハンドラーDLL名称、インポートライブラリ名称の変更、インクルードファイルの変更及びメッセージID番号の変更のみでドライバー移植を完成する事ができます。また、本ボードで追加されたハードウェア機能を採用される場合は、同表後半の明らかに異なる関数名称を付与された関数群を混在させて追加採用する事で実現が可能です。

表4-3 関数一覧

連番	関数名	機能・動作	パラメータ等
【1】	DA781_Open_DASys	ボード、本ハンドラーの初期化	使用ボード数
【2】	DA781_Reset_Board	全ボードのリセット	
【3】	DA781_Set_Trigger	トリガ関連の設定	トリガ源、極性
【4】	DA781_Set_SampMode	波形データ転送モードの設定	バッファ領域
【5】	DA781_Set_Exclk	オプション外部クロック値の設定	外部クロック周波数
【6】	DA781_Set_Clock	クロック源、クロック値の設定	クロック源、分周比
【7】	DA781_Set_Scale	スケール変数の設定	スケール変数データ
【8】	DA781_Write_DLLBuf	DLL 管理バッファへのデータ転送	波形出力データ
【9】	DA781_Write_DirectFifo	ボード内メモリへの直接データ転送	波形出力データ
【10】	DA781_Start_Samp	波形出力スタート	DAデータ数・更新モード
【11】	DA781_Out_Prompt	DA 即時更新出力	DAデータ
【12】	DA781_Get_Status	ボードステータスの取得	ボードステータス
【13】	DA781_Stop_Samp	波形出力の強制停止	
【14】	DA781_Close_DASys	本ハンドラーの終了	
【15】	DA781_Out_Aux	汎用デジタル出力	更新データ
【16】	DA781_Inp_Aux	汎用デジタル入力	
【17】	DA781_Set_Datacode	DA データコードの設定	バイナリ/2 の補数等
【18】	DA781_Sel_Outsig	クロック/SYNC 出力切換え	
【19】	DA781_Clear_Flags	ボードステータスのクリア	
【20】	DA781_Get_Libver	本ハンドラーのバージョン情報取得	
以下の関数は FCDA0112PCI で導入されたハードウェアを制御するためのものです			
【21】	DA0112_Get_DataCode	アナログ回路情報、ハーフフルサイズ取得	
【22】	DA0112_Set_DataCode	アナログ回路情報、ハーフフルサイズ設定	
【23】	DA0112_Get_MemSize	メモリ容量及び空きエリア数の取得	

※ 上でも述べたようにメッセージID番号はマイクロサイエンス製ドライバーとの相互干渉を避ける意味からそれとは異なった番号 (WM_USER + 0x1112) を採用しています。

【1】ボード、本ハンドラーの初期化

int __stdcall DA781_Open_DASys(HWND Owner, int num_board, int *fifo_size);	
引数	Owner:ウィンドーハンドル num_board:使用ボード枚数 fifo_size : オンボードメモリサイズを報告する変数
戻り値	正常終了時 : 0 エラー発生時 : エラーコード (負の値)
機能・動作	プラグアンドプレイで設定された各ボードの I/O アドレス、及び割り込みレベルを認識すると同時にボードリセット、その他ハンドラー内の各種テーブルやデータバッファ等を初期化する。

【2】全ボードのリセット

int __stdcall DA781_Reset_Board(void);	
引数	なし
戻り値	正常終了時 : 0 エラー発生時 : エラーコード (負の値)
機能・動作	全ての制御レジスタは初期状態に戻る。(汎用デジタル出力は変化しない) アナログ出力は共に 0 V に戻る。なお、【1】初期化実行直後には必要ない

【3】トリガ関連の設定

int __stdcall DA781_Set_Trigger(int trig_source, int trig_pol);	
引数	trig_source:トリガ源 (0 : 即トリガ、1 : 外部トリガ) trig_pol:外部トリガ極性 (0 : 負エッジ (↓)、1 : 正エッジ (↑)、2 : 負レベル、3 : 正レベル)
戻り値	正常終了時 : 0 エラー発生時 : エラーコード (負の値)
機能・動作	波形出力スタート条件を設定する。外部トリガ極性でレベルを指定したときは指定されたレベル間だけ波形出力が行われる帯域動作となる。

【4】波形データ転送モードの設定

int __stdcall DA781_Set_SampMode(int trs_mode, int buf_area, int intr_sw);	
引数	trs_mode:波形データ追加転送の起動要因 0 : ノットフル、1 : ノットハーフフル buf_area : 波形データ格納バッファ指定 : 0 : ハンドラー内 Dll の管理するバッファ 1 : ユーザープログラムの管理するバッファ intr_sw:割り込みイベント発生要因指定 0 : 割り込み不使用 1 : 連続サンプリングクロック 2 : 外部割り込み入力信号 (↓) 3 : 外部割り込み入力信号 (↑) 4 : トリガ発生 5 : 指定ブロック数出力終了 6 : 1 ブロック出力開始 7 : ノットハーフフル状態
戻り値	正常終了時 : 0 エラー発生時 : エラーコード (負の値)
機能・動作	波形データ追加転送 (非サイクルモード) 及び波形データバッファに関する条件を設定する。 割り込みイベントは現状未対応

【5】オプション外部クロック値の設定

int __stdcall DA781_Set_Exclk(int exclk_freq);	
引数	exClk_freq: オプションまたは外部クロック源の周波数 (Hz)
戻り値	正常終了時: 0 エラー発生時: エラーコード (負の値)
機能・動作	オプションまたは外部クロック源を使用する場合、その周波数 (Hz) を設定する。クロック設定で分周比を使用する場合若しくは標準搭載の内部クロックを使用する場合は必要ない。

【6】サンプリング（波形出力同期）クロック値の設定

int __stdcall DA781_Set_Clock(int clk_source, int set_mode, int *time_unit, int *clk_period);	
引数	clk_source: クロック源指定 0: 内部クロック源0 (10MHz) 1: 内部クロック源1 (8.192MHz) 2: 内部クロック源2 (オプション) 3: 内部クロック源3 (未使用) 4: 外部クロック源 (立下りエッジ) 5: 外部クロック源 (立上りエッジ) set_mode: サンプリングクロックの指定方法 0: クロック周期の値 1: 分周比 time_unit: クロック周期の単位 0: S 1: mS 2: uS 3: nS clk_period: クロック周期の値、または分周比
戻り値	正常終了時: 0 エラー発生時: エラーコード(負の値)
機能・動作	サンプリングクロック値を設定する。なお設定できない値が指定されたときは設定可能な長いほうの近似値が設定される。また、クロック源を2, 4, 5から選び、クロックを周期の値で指定する場合は前記【5】DA781_Set_Exclkによってクロック源の周波数値を定義しておく必要がある。

【7】スケール変数の設定

int __stdcall DA781_Set_Scale(int board_no, int scale);	
引数	board_no: 空きエリア数を確認するボードの番号 0: 1枚目 (マスター)、1~7: スレーブ scale: スケール変数データ (0~4095)
戻り値	正常終了時: 0 エラー発生時: エラーコード (負の値)
機能・動作	振幅制御 DAC へ制御データを出力する。更新は更新モードによって異なり、即時更新では、データ書き込みと同時に、クロック同期更新モードでは、本出力動作後のクロックに同期して出力が変更される。

【8】DLL 管理バッファへのデータ転送

int __stdcall DA781_Write_DLLBuf(int board_no, int num_samp, int num_data, UINT_PTR *bufptr);	
引数	board_no: 波形データを書き込むボードの番号 0 : 1 枚目 (マスター)、1 ~ 7 : スレーブ num_samp: 対象ボード用の総データ数 (各ボード共通) num_data: 1 ブロック当たりのデータ数 (2 ~ 6 5 5 3 5) bufptr: 波形データバッファのポインター
戻り値	正常終了時 : 0 エラー発生時 : エラーコード (負の値)
機能・動作	ハンドラーが管理する DLL バッファへボード毎の出力データをセットする関数である。データバッファの構成としては、ボード番号0から順に num_data ブロックが n 個の形となっている。

DLL バッファの内部構成は以下の通り

ボード番号0 ブロック番号0	先頭 DA データ 2 番目 DA データ . n 番目 DA データ
ボード番号0 ブロック番号1	先頭 DA データ 2 番目 DA データ . n 番目 DA データ
ボード番号1 ブロック番号0	先頭 DA データ 2 番目 DA データ . n 番目 DA データ
途中省略	
.	
ボード番号m ブロック番号1	先頭 DA データ 2 番目 DA データ . n 番目 DA データ

上表は、ボード枚数m+1 枚、データブロック数2、データ1ブロック=n 個 DA データの場合を示している。

【9】ボード内メモリへの直接データ転送

int __stdcall DA781_Write_DirectFifo(int board_no, int num_data, UINT_PTR *bufptr);	
引数	board_no: 波形データを書き込むボードの番号 0 : 1 枚目 (マスター)、1 ~ 7 : スレーブ num_data: 書き込みデータ数 bufptr: 波形データバッファのポインター
戻り値	正常終了時 : 0 エラー発生時 : エラーコード (負の値)
機能・動作	本ハンドラーのデータバッファを使用せず、ユーザープログラム側から直接ボード内のメモリに、データを転送する。波形出力動作前の場合は FIFO が満杯にならない限度まで書き込みが可能、又出力動作中は転送フラグにより 1 データ或いはハーフフルサイズ以下のデータ数までオーバーランにならない限度まで連続して書き込みが可能である。

【10】波形出力スタート

int __stdcall DA781_Start_Samp(WORD num_data, WORD num_block, int upd_mode);	
引数	num_data: 1 ブロック当たりのデータ数 (2 ~ 6 5 5 3 5) num_block: 出力ブロック数 (1 ~ 6 5 5 3 5) upd_mode: 更新モード 0 : サイクル、num_data 分のデータを停止まで連続出力 1 : サイクル、num_data × num_block 分のデータを出力 2 : 非サイクル、num_data × num_block 分のデータを出力 3 : 非サイクル、強制停止まで連続出力 (FIFO へ出力データを連続供給する必要がある)
戻り値	正常終了時 : 0 エラー発生時 : エラーコード (負の値)
機能・動作	サンプリングをスタートさせる関数。外部トリガ指定の場合は、トリガ待ち状態となる。

【11】DA 即時更新出力

int __stdcall DA781_Out_Prompt(int board_no, int upd_prompt);	
引数	board_no: 更新出力する対象ボード番号 0 : マスター、1 ~ 7 スレーブ upd_prompt: DA 出力データ (0 ~ 4 0 9 5)
戻り値	正常終了時 : 0 エラー発生時 : エラーコード (負の値)
機能・動作	Upd_Prompt データで DA 出力が即時更新される

【12】ボードステータスの取得

int __stdcall DA781_Get_Status(int board_no, WORD *cnt0_data, WORD *cnt1_block, int *eos, int *intr_req, int *upd_err, int status[]);	
引数	board_no: ステータスを取得する対象ボードの番号 0 : マスター、1 ~ 7 : スレーブ cnt0_data: 1 ブロック中の出力データ数カウンタ値 cnt1_block: 出力ブロック数カウンタの現在値 eos: DA 出力終了 1 : 終了 / 0 : 出力中または開始前 intr_req: 割り込み要求発生 1 : 発生 / 0 : 未発生 upd_err: 更新出力エラー 2 : オーバーラン発生、1 : アンダーラン発生、0 : 未発生 status[0]: ボードステータス 1 1 ワード生データ
戻り値	正常終了時 : 0 エラー発生時 : エラーコード (負の値)
機能・動作	動作進行状況または終了検出を行うための関数。なおフラグ関連ステータスは一旦セットされると【1】初期化、或いは【2】ボードリセットまたは【20】フラグクリア関数で解除するまで保持される

【13】波形出力の強制停止

int __stdcall DA781_Stop_Samp(void);	
引数	なし
戻り値	正常終了時 : 0 エラー発生時 : エラーコード (負の値)
機能・動作	全チャンネル (全ボード) のサンプリング動作を強制的に停止する関数。

【14】本ハンドラーの終了

int __stdcall DA781_Close_DASys(void);	
引数	なし
戻り値	正常終了時：0 エラー発生時：エラーコード（負の値）
機能動作	本ハンドラーの終了。確保したメモリ領域の開放等開始前の状態に戻す。DA出力最終状態を維持するため、ボードのリセットは行わない。

【15】汎用デジタル出力更新

int __stdcall DA781_Out_Aux(int board_no, int out_data);	
引数	board_no:更新対象ボード番号 0：マスター、1～7：スレーブ out_data: 1ビット汎用デジタル出力データ
戻り値	正常終了時：0 エラー発生時：エラーコード（負の値）
機能・動作	TTLレベルの1ビット汎用デジタル出力を更新する。当出力ポートは電源投入時“0”となるが、【1】初期設定、【2】ボードリセット、【14】ハンドラー終了等を実行しても変化しない。

【16】汎用デジタル入力

int __stdcall DA781_Inp_Aux(int board_no);	
引数	board_no:汎用デジタル入力を読み取るボードの番号 0：マスター、1～7：スレーブ
戻り値	正常終了時：1ビット汎用デジタル入力データ エラー発生時：エラーコード（負の値）
機能・動作	TTLレベルの1ビット汎用デジタル入力の値を取得する

【17】DAデータコード、出力モード等の設定

int __stdcall DA781_Set_Datacode(int data_code);	
引数	data_code:DAデータコード切換え（1：2の補数、0：オフセットバイナリ）
戻り値	正常終了時：0 エラー発生時：エラーコード（負の値）
機能・動作	本ボードの動作状態を規定する。

【18】クロック／SYNC出力切換え

int __stdcall DA781_Sel_Outsig(int out_sig[]);	
引数	out_sig[]:クロック／SYNC出力選択 0：クロック、1：SYNC []内添え字はボード番号と対応させる。複数ボード使用時のマスターではその値を0とする必要がある
戻り値	正常終了時：0 エラー発生時：エラーコード（負の値）
機能・動作	外部へのクロック信号出力を必要に応じてSYNC信号（各ブロック出力開始タイミング信号）に変更することができる。但し複数枚使用時のマスターについては、スレーブとの関係でクロック信号を出力するため注意が必要である。

【19】波形出力条件の再設定（前回と同一）

int __stdcall DA781_Clear_Flags(int set_mode);	
引数	set_mode: D A出力条件再設定の有無 0 : フラグクリアのみ、1 : フラグクリア後 D A出力条件を再設定する
戻り値	正常終了時 : 0 エラー発生時 : エラーコード (負の値)
機能・動作	ステータスレジスタの各フラグをクリアし、(set_mode が 1 の場合) サンプリング条件を前回と同じ条件として再設定する。

【20】本ハンドラーのバージョン情報取得

int __stdcall DA781_Get_Libver(int ver);	
引数	Ver: バージョン番号の指定値 0 : メジャーバージョン番号+マイナーバージョン番号 1 : メジャーバージョン番号 2 : マイナーバージョン番号
戻り値	正常終了時 : 0 エラー発生時 : エラーコード (負の値)
機能・動作	本ハンドラーのバージョン番号を取得する。

【21】DA データコード、出力モード等の取得

int __stdcall DA0112_Get_DataCode (WORD board_no, WORD *son, WORD *aorb, WORD *ref_sel, WORD *out_mode, WORD *data_code);	
引数	board_no: 制御対象ボードの番号 0 : マスター、1～7 : スレーブ son: D A出力バッファオンオフ制御状態取得変数 (1 : オン、0 : オフ) aorb: 出力モード指定状態取得変数 (1 : A モード、0 : B モード) ref_sel: 基準電源切換え状態取得変数 (1 : 外部基準電源、0 : 内部基準電源) out_mode: 出力モード切換え状態取得変数 (1 : ユニポーラ、0 : バイポーラ) data_code: D Aデータコード切換え状態取得変数 (1 : 2 の補数、0 : オフセットバイナリ)
戻り値	正常終了時 : 0 エラー発生時 : エラーコード (負の値)
機能・動作	本ボードの動作状態を引数夫々に引き渡す。アプリケーション側では、この情報を元に D Aボードの動作状態を確認・変更する事ができる。data_code 変数については、ユーザーの利便を図り本関数で状態を取得できるよう考慮している。

【22】DA データコード、出力モード等の設定

int __stdcall DA0112_Set_DataCode (WORD board_no, WORD *son, WORD *aorb, WORD *ref_sel, WORD *out_mode);	
引数	board_no: 制御対象ボードの番号 0 : マスター、1～7 : スレーブ son: D A出力バッファオンオフ状態制御変数 (1 : オン、0 : オフ) aorb: 出力モード状態制御変数 (1 : A モード、0 : B モード) ref_sel: 基準電源切換え状態制御変数 (1 : 外部基準電源、0 : 内部基準電源) out_mode: 出力モード切換え状態制御変数 (1 : ユニポーラ、0 : バイポーラ)
戻り値	正常終了時 : 新ハーフフルサイズ (ワード) エラー発生時 : エラーコード (負の値)
機能・動作	本ボードの動作状態を引数夫々により変更する。

【23】 空きメモリエリア数取得

int __stdcall DA0112_Get_MemSize (int board_no, int Mem_Size, int Free_Area);	
引数	board_no:空きエリア数を確認するボードの番号 0 : 1 枚目 (マスター)、1 ~ 7 : スレーブ Mem_Size:搭載メモリサイズを返す変数 (単位:ワード) Free_Area:空きエリア数を返す変数 (単位:ワード)
戻り値	正常終了時: 0 エラー発生時: エラーコード (負の値)
機能・動作	本ボード内メモリサイズ及びその空きエリア数を取得する関数である

4-7 エラーコード

戻り値	不具合の内容	適用関数等
- 1	必要な PCI ライブラリが見つからない	DA781_Open_ADSSys
- 2	FCDA0112PCI ボードが見つからなかった	DA781_Open_ADSSys, DA781_Reset_Board
- 3	マスターボードが見つからなかった	DA781_Open_ADSSys
- 4	SW-BN スイッチが重複している	DA781_Open_ADSSys
- 5	指定されたボード数と実際の枚数が異なっていた	DA781_Open_ADSSys
- 6	複数ボード間でハーフフルサイズが異なっていた	DA781_Open_DASys
- 7	指定されたユニット番号のボードが存在しない	各関数
- 8	オープンされているボードがない	各関数
- 9	スタートしているボードがない	各関数
- 10	既にサンプリング中	各関数
- 11	転送フラグ指定エラー	DA781_Set_SampMode, DA781_Write_DirectFifo
- 12	バッファ指定エラー	DA781_Set_SampMode
- 13	データコード指定が間違っている	DA781_Set_Datacode
- 14	トリガ選択エラー	DA781_Set_Trigger
- 15	トリガ源選択エラー	DA781_Set_Trigger
- 16	外部クロック値設定エラー	DA781_Set_Exclk, DA781_Set_Clock
- 17	クロックソース指定が異常	DA781_Set_Clock
- 18	クロック周期指定が異常	DA781_Set_Clock
- 19	スキャン周期指定が仕様範囲外	DA781_Set_Clock
- 20	クロック指定方法が間違っている	DA781_Set_Clock
- 21	データブロック長が異常	DA781_Start_Samp
- 22	データブロック数が異常	DA781_Start_Samp
- 23	更新モードがおかしい	DA781_Start_Samp
- 24	Dll バッファサイズがおかしい	DA781_Write_DLLBuf, DA781_Write_DirectFifo
- 25	メモリ確保に失敗した	DA781_Write_DLLBuf, DA781_Write_DirectFifo
- 26	Dll バッファへの書き込みがエラーになった	DA781_Write_DLLBuf
- 27	メモリ容量を超えて書き込みを行った	DA781_Write_DirectFifo
- 28	FIFO メモリに余裕がない	DA781_Write_DirectFifo
- 29	クロック・SYNC 切り替えコードエラー	DA781_Sel_Outsig
- 30	バージョン番号指定エラー	DA781_Get_Libver
- 31	出力バッファオンオフ制御設定値が異常	DA0112_Set_DataCode
- 32	A/B モード選択制御設定値が異常	DA0112_Set_DataCode
- 33	基準電源内部外部切り替え制御設定値が異常	DA0112_Set_DataCode
- 34	ユニポーラ・バイポーラ切り替え制御設定値が異常	DA0112_Set_DataCode
- 35	ハーフフルサイズ設定値が異常	DA0112_Set_DataCode
- 36	搭載メモリサイズ設定値が異常	DA0112_Get_MemSize
- 37	Set_Mode 変数が範囲外だった	DA781_Clear_Flags
- 38	循環モードでユーザーバッファを指定した	DA781_Start_Samp
- 39	データロス等のランタイムエラーが発生した	DA781_Write_DirectFifo

第5章 汎用デバイスドライバ仕様書

本仕様書は、フレックスコア製 PCI ボードを制御するために必要となる汎用デバイスドライバの内容について記述したものです。本デバイスドライバは、最大で 16 台までの PCI ボードを同時に制御する事が可能ですが、基本的な仕組みは、ドライバオープン及びドライバスタートから始まり、夫々の PCI ボードリソースを、I/O 制御レジスタへのアクセスにより制御し、全ての制御が終了した段階でドライバストップ、更にドライバクローズ処理を行って制御全てを終了するところまでを行います。

基本的なリソース制御については、バイト制御、バイトブロック制御、ワード制御、ワードブロック制御、更にダブルワード制御及びダブルワードブロック制御の 12 種類（各々リード・ライト処理がある）が定義されています。実際には、本仕様書脱稿の段階でダブルワード制御を行うレジスタを装備したボードは存在しませんが、将来的にそのようなボードが成立する可能性も充分考えられるため、ハンドラー仕様書として内容を整備しています。

本ドライバは、Windows7 等での PCI ボードインストールとの関係上、対応する専用ドライバインストール時に共にインストールされるよう設定されています。（即ち専用のインストーラーは存在しません。）

また、本ドライバは、上記の如く単純な I/O レジスタアクセスのみから構成されている為、基本的に無サポートとします。また全てのデータ処理（バイト/ワード/ダブルワード：リード/ライト）については、クリティカルセクションによって排他処理が保証されています。

※ クリティカルセクションについては、特にマルチコア CPU 等の環境で本ドライバを使用する際に必要となるものですが、ここではその詳細については記述しません。詳細については、マイクロソフト製ビジュアルスタジオ等に詳細な記述があるのでこれを参照してください。

ハンドラー及び関連ファイル名称一覧表

OS 及びビット幅	Windows7 32 ビット	Windows7 64 ビット	WindowsXP 32 ビット
ドライバ名称	PORT7_X86.Dll	PORT7_X64.Dll	PORT7_X86.Dll
インポートライブラリ名称	PORT7_X86.Lib	PORT7_X64.Lib	PORT7_X86.Lib
関数インデックス	PORT7.H	PORT7.H	PORT7.H

上記ドライバ 3 種類の内、WindowsXP32 ビット用ドライバについては、システム移行動作確認用としてのみ使用する事を前提として公開しています。このドライバは、WindowsXP システムで動作していたアプリを Windows7 へ移行する際に、その操作を段階的に（つまり OS の変更を行うこと無しに）新しいドライバでの動作評価を行うことができるよう用意されているものです。（ハンドラーの内容としては Windows7 32 ビット版と全く同じ構成になっています。）

以下に記述する仕様は、ある固有ベンダー ID に従った PCI ボード群に対する汎用デバイスドライバの仕様書です。ベンダー ID は固有の値（現状 0 x 1 3 F D）でなければなりません。

【1】ドライバーオープン

int __stdcall Open_Driver(int *device_id);	
引数	device_id:各ボードのデバイス ID をセットする配列 (ユーザー側にて確保:最大 16 個)
戻り値	正常終了時: 0 エラー発生時: エラーコード (負の値)
機能・動作	ハンドラー内部の初期化を行い、システム内で検出されたそれぞれのボードのデバイス ID を報告する。

【2】ドライバースタート

int __stdcall Start_Driver(int board_no, int base_no, int *resorce_no);	
引数	board_no: ドライバーオープンで報告された、デバイス ID を格納している配列の要素番号であり、尚且つシステム内で検索され発見された該当ボードの発見順序を表す変数である base_no: I O リソース番号 (2 固定) resource_no: 当汎用ハンドラー内で確定したリソースを指し示す要素番号
戻り値	正常終了時: 0 エラー: エラーコード (負の値)
機能・動作	ドライバーオープンで検出されたボード群の内 board_no で指定したボードと通信を開始する。

【3】バイトリード

int __stdcall ReadByte(int resource_no, int offset);	
引数	resource_no: スタートドライバー関数から返されたボードリソースインデックス offset: 操作対象レジスタのオフセットアドレス
戻り値	正常終了時: 読み込んだバイト値 エラー発生時: エラーコード (負の値)
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタの値を読み込みユーザーへ返す関数 (戻り値=バイト)

【4】バイトブロックリード

int __stdcall ReadByteBlock(int resource_no, int offset, int count, BYTE *pbuf);	
引数	resource_no: スタートドライバー関数から返されたボードリソースインデックス offset: 操作対象レジスタのオフセットアドレス count: 読み込みを行うバイト数 pbuf: 読み込んだレジスタからの値を保存する先頭アドレス (ポインタ)
戻り値	正常終了時: 0 エラー発生時: エラーコード (負の値)
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタの値を count 回数読み込みユーザーが用意したデータバッファ (pbuf) へ返す関数である。pbuf データバッファのサイズは count (バイト) 必要である。

【5】ワードリード

int __stdcall ReadWord(int resource_no, int offset);	
引数	resource_no: スタートドライバ関数から返されたボードリソースインデックス offset: 操作対象レジスタのオフセットアドレス
戻り値	正常終了時: 読み込んだワード値 エラー発生時: エラーコード (負の値)
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタの値を読み込みユーザーへ返す関数

【6】ワードブロックリード

int __stdcall ReadWordBlock(int resource_no, int offset, int count, WORD *pbuf);	
引数	resource_no: スタートドライバ関数から返されたボードリソースインデックス offset: 操作対象レジスタのオフセットアドレス count: 読み込みを行うワード数 pbuf: 読み込んだレジスタからの値を保存する先頭アドレス (ポインタ)
戻り値	正常終了時: 0 エラー発生時: エラーコード (負の値)
機能・動作	board_no で示されるボードの offset アドレスに存在するレジスタの値を count 回数読み込みユーザーが用意したデータバッファ (ret_value) へ返す関数である。Ret_value データバッファのサイズは count×2 (バイト) 必要である。

【7】ダブルワードリード

int __stdcall ReadDWord(int resource_no, int offset);	
引数	resource_no: スタートドライバ関数から返されたボードリソースインデックス offset: 操作対象レジスタのオフセットアドレス
戻り値	正常終了時: 読み込んだダブルワード値 エラー発生時: エラーコード (負の値)
機能・動作	board_no で示されるボードの offset アドレスに存在するレジスタの値を読み込みユーザーへ返す関数

【8】ダブルワードブロックリード

int __stdcall ReadDWordBlock(int resource_no, int offset, int count, UINT32 *pbuf);	
引数	resource_no: スタートドライバ関数から返されたボードリソースインデックス offset: 操作対象レジスタのオフセットアドレス count: 読み込みを行うワード数 pbuf: 読み込んだレジスタからの値を保存する先頭アドレス (ポインタ)
戻り値	正常終了時: 0 エラー発生時: エラーコード (負の値)
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタの値を count 回数読み込みユーザーが用意したデータバッファ (pbuf) へ返す関数である。pbuf データバッファのサイズは count×4 必要である。

【9】バイトライト

int __stdcall WriteByte(int resource_no, int offset, BYTE data);	
引数	resource_no: スタートドライバ関数から返されたボードリソースインデックス offset: 操作対象レジスタのオフセットアドレス data: レジスタへ書き込む値
戻り値	正常終了時: 0 エラー発生時: エラーコード (負の値)
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタへ data を書き込む関数

【10】バイトブロックライト

int __stdcall WriteByteBlock(int resource_no, int offset, int count, BYTE *pbuf);	
引数	resource_no: スタートドライバ関数から返されたボードリソースインデックス offset: 操作対象レジスタのオフセットアドレス count: 読み込みを行うワード数 pbuf: 読み込んだレジスタからの値を保存する先頭アドレス (ポインタ)
戻り値	正常終了時: 0 エラー発生時: エラーコード (負の値)
機能・動作	board_no で示されるボードの offset アドレスに存在するレジスタへ pbuf バッファから count 回数書き込みを行う関数である。Pbuf データバッファのサイズは count (バイト) 必要である。

【11】ワードライト

int __stdcall WriteWord(int resource_no, int offset, WORD data);	
引数	resource_no: スタートドライバ関数から返されたボードリソースインデックス offset: 操作対象レジスタのオフセットアドレス data: レジスタへ書き込む値
戻り値	正常終了時: 0 エラー発生時: エラーコード (負の値)
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタへ data に保存されているデータを書き込む関数

【12】ワードブロックライト

int __stdcall WriteWordBlock(int resource_no, int offset, int count, WORD *pbuf);	
引数	resource_no: スタートドライバ関数から返されたボードリソースインデックス offset: 操作対象レジスタのオフセットアドレス count: 書き込みデータ数 pbuf: レジスタへ書き込む値を保存しているアドレス (ポインタ)
戻り値	正常終了時: 0 エラー発生時: エラーコード (負の値)
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタへ pbuf に保存されているデータを書き込む関数。pbuf データバッファのサイズは count × 2 (バイト) 必要である。

【13】ダブルワードライト

int __stdcall WriteDWord(int resource_no, int offset, UINT32 data);	
引数	resource_no: スタートドライバ関数から返されたボードリソースインデックス offset: 操作対象レジスタのオフセットアドレス data: レジスタへ書き込む値
戻り値	正常終了時: 0 エラー発生時: エラーコード (負の値)
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタへ data に保存されているデータを書き込む関数

【14】ダブルワードブロックライト

int __stdcall WriteDWordBlock(int resource_no, int offset, int count, UINT32 *pbuf);	
引数	resource_no: スタートドライバ関数から返されたボードリソースインデックス offset: 操作対象レジスタのオフセットアドレス count: 書き込みデータ数 pbuf: レジスタへ書き込む値を保存しているアドレス (ポインタ)
戻り値	正常終了時: 0 エラー発生時: エラーコード (負の値)
機能・動作	resource_no で示されるボードの offset アドレスに存在するレジスタへ pbuf に保存されているデータを書き込む関数。pbuf データバッファのサイズは count × 4 (バイト) 必要である。

【15】ドライバーストップ

int __stdcall Stop_Driver(int resource_no);	
引数	resource_no: ドライバースタート関数から返されたボードリソースインデックス
戻り値	正常終了時: 0 エラー発生時: エラーコード (負の値)
機能・動作	全てのユニットで使用していたメモリリソースを開放し通信を終了する。

【16】ドライバークローズ

int __stdcall Close_Driver(void);	
引数	なし
戻り値	正常終了時: 0 エラー発生時: エラーコード (負の値)
機能・動作	全てのボードについてハンドルを開放する。

エラーコード

戻り値	内容
－ 1	必要な PCI ライブラリが見つからない
－ 2	ライブラリのデバッグオプションが無効だった
－ 3	PCI ボードが見つからなかった
－ 4	ボード番号指定がシステム上の枚数を上回っている
－ 5	オープンしていないボードをクローズしようとした
－ 6	定義されていないベースナンバーを参照しようとした
－ 7	オープンしていないボードに対してアクセスが行われた
－ 8	リソース範囲不適
－ 9	内部メモリ確保エラー

第6章 保守、その他

6-1 故障・トラブル等の原因と対処

本ボードは、DOS/Vパソコンで全数検査の上出荷されています。
お手元での動作確認方法は1-6項に記されています。動作に不具合がある時は、以下の諸点を再点検して下さい。
それでも不明なときは、システム構成等をご記入の上、弊社品質管理部まで御連絡下さい。情報伝達手段としてfax及びEmailを準備しておりますが、ご利用しやすい方のお手元をお選びの上、御連絡願います。

一般的に言って、応答速度を求められるときはEmailが、複雑な情報を含む場合はfaxがお勧めだと思います。

再点検、確認ポイント

ボードID

単体で動作させる時、この番号が0以外の設定になっているとハンドラーがユニットを認識しません。
実際に起こり得る局面としては、マスタースレーブ接続で使用されていたスレーブユニットを、単独で使用する等の局面が考えられます。

デジタル入出力

本ボードのデジタル入力に接続できる信号源はTTL(LS,CMOS等の5V電源動作素子)に限ります。
それ以外の電源電圧を用いている信号源を接続すると、殆ど一瞬のうちに入力素子が破壊します。
出力についてもTTL信号源以外の信号源に接続することは、即出力素子の破壊を招きます。
本ボードで使用しているデジタル入出力素子の絶対最大定格は-0.6V～+7.0Vとなっています。

アナログ入力

本ボードの過電圧入力保護は±3.5Vが最大です。そのため一瞬であってもこの電圧範囲を超える電圧が印加されるとアナログ入力回路素子の破壊につながります。また、アナログ回路は、デジタル回路と異なり取り扱う電圧そのものが高く、更に異常電圧が印加された場合に回路各部へ伝播していく可能性も高いため、異常電圧による被害はデジタル回路に比べて大きくなりがちです。

システム相互間のグランド電位差にも注意が必要です。特に様々な機器からの入力を接続する機会が多い場合、この原因による回路破壊も時折見られます。予防策としては、夫々の回路を接続する前に、機器間のグランド電位差をテスターなどで確認しておく事があげられます。一般的には交流的な電位差がある場合が多いと考えられますが、直流的な電位差についても確認が必要です。

更に、本ボードのグランド電位は登載されているパソコンのグランド電位と共通であるという点にも注意が必要です。

6-2 修理のときは

本製品使用上の質問・トラブル対応・故障修理などは入手経路の如何にかかわらず、弊社宛に直接御相談下さい。
商社等を経由されても弊社として問題はありますが、情報交換の密度、修理完了までの時間などの面で多少不利になるのではないかと思います。

また、導入当初からの不具合については、誤解や情報不足による事が多いので、事前にご相談下さい。

無償修理と有償修理の場合分けにつきましては以下の通りとなります。

納入後1年間は自然故障、及び弊社製造上の問題に起因した事が明らかな故障製品に対して無償修理を行います。

但し保証期間中であっても、次の場合は有償修理となります。

- ア 取扱上の不注意、誤用による故障および損傷
- イ 弊社以外での修理、改造、分解掃除等による故障および損傷
- ウ 泥、砂、水などのかぶり、落下、衝撃等が原因で発生した故障および損傷
- エ 火災、地震、水害、落雷その他の天変地異、公害や異常電圧による故障および損傷
- オ 保管上の不備（高温多湿の場所等）や手入れの不備による故障
- カ 接続している他の機器に起因して故障が生じた場合
- キ その他使用者側の責に帰する故障

また、不良要因が再現されなかった場合につきましては、着払いにより製品を返却させて頂く場合もあります

修理は宅配便によるセンドバックで行います。尚運賃は互いに発送する側が負担するものとします。
出張修理は行っておりません。簡単な故障であれば一週間程度で修理可能ですが、故障状況によっては更に日数を要する事もあります。
また、有償修理の場合の修理料金については、簡単な故障で基本料金¥10,000円+部品代とお考え下さい（事務手数料を含む）。修理費用限度額がある場合は、お申し出頂ければ超過する事が明確になった時点で御連絡いたします。

修理品送付先

〒301-0853

茨城県龍ヶ崎市松ヶ丘3丁目18番地3

合同会社 フレックスコア

品質管理部 林

E-mail: support@flexcore.jp

Fax:050-3488-3354

6-3 アナログ出力範囲の再調整

1-6項に示すような形で動作確認を行い、出力範囲やオフセットの変化が認められるときは再調整が必要です。アナログ回路は経年環境変化に対する保守を定期的に行うことが望ましく、夏冬の使用環境・周囲温度に差がある場合は季節単位、通年安定した使用環境の場合は1～2年に1度は校正することが理想的です。

再調整の方法・手順を以下に記しますが、被調整対象のFCDA0112PCI-16MWの他に、高精度の基準電源と高精度の電圧計、各入力を個別に切換える事ができるスイッチボックス及びWindow XP/Vistaがインストールされたパソコンを準備する必要があります。

6-4 外部制御信号、タイミング等